# AFRL-SN-WP-TR-2005-1108


# DARPA INTEGRATED SENSING AND PROCESSING (ISP) PROGRAM
## Approximation Methods for Markov Decision Problems in Sensor Management

**Michael K. Schneider, Ph.D.**
**Angelia Nedich, Ph.D.**
**Prof. David Castanon**
**Bob Washburn, Ph.D.**

**BAE Systems**
**Advanced Information Technology**
**6 New England Executive Park**
**Burlington, MA 01803**

**JUNE 2006**

**Final Report for 01 July 2002 – 30 June 2006**

Approved for public release; distribution is limited.

**STINFO COPY**

Appendices 2 through 4, resulting from Air Force contract number F33615-02-C-1197, have been submitted for publication in various conference proceedings. If published, the United States has for itself and others acting on its behalf an unlimited, paid-up, nonexclusive, irrevocable, worldwide license. 

**SENSORS DIRECTORATE**
**AIR FORCE RESEARCH LABORATORY**
**AIR FORCE MATERIEL COMMAND**
**WRIGHT-PATTERSON AFB, OH 45433-7320**

# NOTICE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory Wright Site (AFRL/WS) Public Affairs Office (PAO) and is releasable to the National Technical Information Service (NTIS). It will be available to the general public, including foreign nationals.

PAO Case Number: AFRL/WS-06-0157, 19 Jan 2006

THIS TECHNICAL REPORT IS APPROVED FOR PUBLICATION.

//Signature//

Alan D. Kerrick, Project Engineer
RF Systems and Analysis Branch
RF Technology Division

//Signature//

Keith W. Loree, Branch Chief
RF Systems and Analysis Branch
RF Technology Division

//Signature//

Timothy R. Poth, Major, USAF
Deputy Division Chief
RF Sensor Technology Division

This report is published in the interest of scientific and technical information exchange and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE *(DD-MM-YY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| June 2006 | Final | 07/01/2002 – 06/30/2006 |

**4. TITLE AND SUBTITLE**
DARPA INTEGRATED SENSING AND PROCESSING (ISP) PROGRAM
Approximation Methods for Markov Decision Problems in Sensor Management

**5a. CONTRACT NUMBER**
F33615-02-C-1197

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**
69199F

**6. AUTHOR(S)**
Michael K. Schneider, Ph.D.
Angelia Nedich, Ph.D.
Prof. David Castanon
Bob Washburn, Ph.D.

**5d. PROJECT NUMBER**
ARPS

**5e. TASK NUMBER**
NR

**5f. WORK UNIT NUMBER**
0U

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
BAE Systems
Advanced Information Technology
6 New England Executive Park
Burlington, MA 01803

**8. PERFORMING ORGANIZATION REPORT NUMBER**
TR-1620

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Sensors Directorate
Air Force Research Laboratory
Air Force Materiel Command
Wright-Patterson AFB, OH 45433-7320

DARPA/DSO
3701 N. Fairfax Dr.
Arlington, VA 22203

**10. SPONSORING/MONITORING AGENCY ACRONYM(S)**
AFRL-SN-WP

**11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)**
AFRL-SN-WP-TR-2005-1108

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**13. SUPPLEMENTARY NOTES**
Report contains color. PAO Case Number: AFRL/WS-06-0157, 19 Jan 2006.
Appendices 2 through 4, resulting from Air Force contract number F33615-02-C-1197, have been submitted for publication in various conference proceedings. If published, the United States has for itself and others acting on its behalf an unlimited, paid-up, nonexclusive, irrevocable, worldwide license. Any other form of use is subject to copyright restrictions.

**14. ABSTRACT**
This work addresses problems of sensor resource management (SRM) in which one or more sensors obtain measurements of the state of one or more targets. For example, an airborne radar may be attempting to track several ground targets, which are sometimes stationary (requiring a synthetic aperture radar mode) and sometimes moving (requiring a ground moving target indication radar mode). The challenge is to schedule the radar modes as the scenario evolves. Such problems can generally be formulated as partially observable Markov decision processes (POMDPs), which can express essential characteristics of the SRM problem such as uncertainty and dynamics. This work emphasizes a farsighted approach; the highest long-term payoff may not be generated by the action providing the highest immediate payoff. Accomplishments of this effort include the establishment of a boundary on optimal SRM performance, analysis of farsighted SRM strategies for controlling a multimode sensor, and the derivation of a novel set of sufficient conditions for optimality in Markov decision processes.

**15. SUBJECT TERMS**
integrated sensing and processing, sensor resource management, partially observable Markov decision processes, farsighted sensor management, radar

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT: | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON (Monitor) |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Alan D. Kerrick |
| Unclassified | Unclassified | Unclassified | SAR | 120 | 19b. TELEPHONE NUMBER *(Include Area Code)* (937) 255-6427, ext. 4343 |

**Standard Form 298 (Rev. 8-98)**
Prescribed by ANSI Std. Z39-18

**TABLE OF CONTENTS**

# 1 LIST OF FIGURES

## 2 LIST OF TABLES

# 3 EXECUTIVE SUMMARY

## 3.1 PROGRAM OVERVIEW

The BAE SYSTEMS Advanced Information Technology (BAE AIT) Integrated Sensing and Processing (ISP) project was analyzing and developing algorithms to solve the problem of closed-loop sensor resource management (SRM), as illustrated in Figure 1. These problems consist of managing one or more sensors to obtain measurements of the state of one or more targets. The measurements are then fused to estimate the states of the targets. The process of fusing the measurements reduces the uncertainty in the measurements and combines measurements to provide complementary information. The measurement collection can be managed at several levels, but it is natural to group these into three categories: collection management, sensor management, and dwell management. A collection manager controls from what locations measurements are made by placing the sensors in those locations. A sensor manager controls what is measured by determining which sensors to use and by adjusting coarse control parameters such as pointing angle and operating mode on an individual sensor. A dwell manager controls how a measurement is made by adjusting sensor characteristics such as transmitted power on an active sensor. The focus of the BAE AIT project was on sensor management.

The potential applications of sensor management are numerous and include ones in the fields of computer network security; environmental monitoring, and air-to-ground intelligence, surveillance, and reconnaissance (ISR).

- In computer network security applications [1], the objective is to monitor the state of a machine to determine if it is being attacked or has been accessed by an intruder. Multiple processes are operating on the computer simultaneously, and software sensors can monitor the activity of each of these processes as well as the aggregate state of the machine. The potential exists to apply sensor management algorithms to dynamically control where and how measurements are made in the system to optimize the detection process.
- In environmental monitoring applications, one is sensing the environment to determine the distribution of one or more environmental resources or pollutants. Potential sensor locations may be predetermined, and then the sensor management problem is one of determining which sensor sites to use or how many to use. An example would be selecting among candidate well sites, determining the ones to use for characterizing the spread of a pollutant underground.
- In air-to-ground ISR applications, the objective is to find, track the position of, and classify ground targets with airborne sensors. The sensors can be steered to look at different areas on the ground. For a fixed sensor platform route, the SRM problem is then one of determining where to collect measurements on the ground.

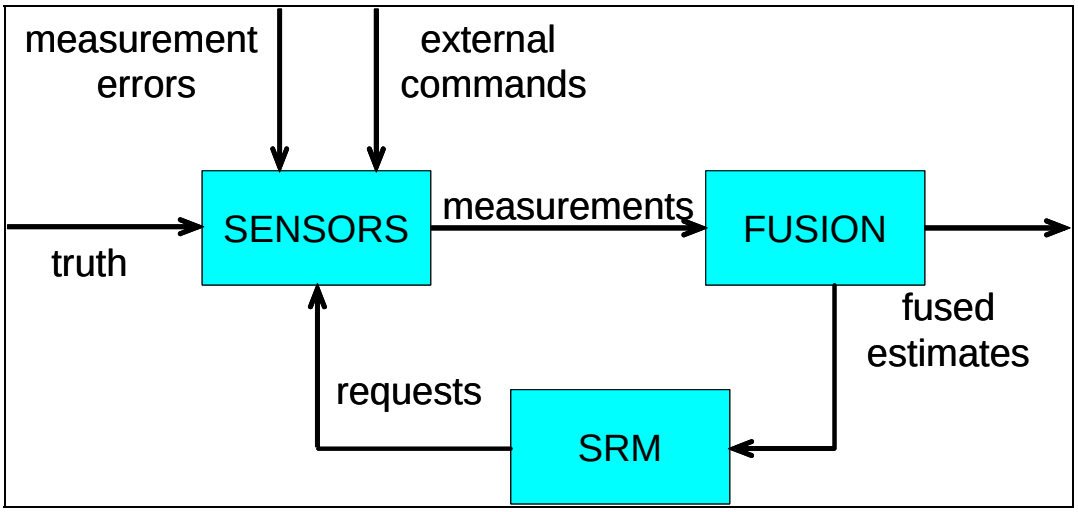


**Figure 1: BAE AIT's ISP project focus on problems of SRM, in which sensors are dynamically managed in closed loop to improve the quality of the data provided to the fusion system**

Our objective in analyzing the SRM problem is to develop insights applicable to a broad class of applications by developing formulations in a general mathematical framework. In particular, SRM problems can generally be formulated as partially observable Markov decision processes (POMDPs). This is a suitable framework because it can express essential characteristics of the SRM problem, such as uncertainty and dynamics. Uncertainty is present in the SRM problems because the data being collected is noisy. Thus, the estimates of targets' true states are always uncertain, and, consequently, the predictions of measurements that would result from future sensor tasks are uncertain. Dynamics are present in SRM problems because one's estimates of true target states are constantly changing. This occurs for two reasons. One is that the true states of the targets are changing. The other is that new data is being collected over time and fused to form new estimates of target states. The uncertainty and dynamics can be modeled in the POMDP framework as a function of the sensor control policy. It dictates what sensor control actions are taken in response to different estimates of target states. The POMDP framework is rich enough to model uncertainties, dynamics and control options in a broad class of SRM problems.

Given a POMDP formulation of a SRM problem including the objective criteria as a function of the data collected, dynamic programming can be used to compute an optimal sensor policy. The dynamic programming algorithm essentially enumerates all possible control actions and outcomes starting from a given collection of historical data and selects the sequence of controls that optimizes the expected outcome as measured by the objective criterion. Dynamic programming is generally difficult to implement when the size of the problem is large. In SRM, the size of the problem is dictated by the set of control options, the set of measurement outcomes resulting from a sensor control, and the set of possible probability distributions of target states. The sets are generally large and, when they are discrete, scale exponentially with the number of targets. As a result, computing the optimal sensor control policy with dynamic programming is difficult and beyond resources of current and foreseeable computers.

Our problem was to analyze a class of SRM problems and develop algorithms for computing control policies which are computationally efficient and near optimal. Although POMDPs arising in SRM problems are large enough that a standard dynamic programming algorithm will be too computationally intensive, the SRM problems have considerable structure that can be exploited to compute good sensor control policies. In particular, the targets in SRM problems have states $x_i$, $i=1,\ldots,n$ that are generally independent. Thus, one would expect that the computation of a sensor policy could be decomposed into a set of computations performed independently on each individual target. However, the sensor control at a particular time is not a function of the target states but of the collection of information on each target $I_j(t)=\{y_j(t_y):t_y<t\}$ for $j=1,\ldots,n$ where $y_j(t_y)$ is a measurement made of target $j$ at time $t_y$. Moreover, the information collections $I_j(t)$ at a particular time are not independent even if the individual measurements $y_j$ of the different targets are independent. This is a consequence of the measurements in each of the collections $I_j(t)$ being selected by the sensor control policy. The dependence among the information collections complicates the structure of the problem. Our challenge is to exploit the structure that is present to find computable, near-optimal sensor control policies.

Many past approaches to developing computable sensor control policies have focused on myopic approaches. Myopic approaches select sensor control actions based on their immediate expected benefits evaluated over the duration of a single sensor action. These policies are generally computationally efficient. However, they do not account for some key aspects of the problem. In particular, they do not account for dynamics in the problem other than the immediate state changes resulting from the next control action. Thus, myopic policies may take an action at a point in time that has immediate benefits but would be better to postpone to a more opportunistic time in the future given the dynamics of the problem. Moreover, myopic policies will generally only be effective if the reward resulting from an action is immediate. If the rewards are not realized until the future, then myopic policies will not work.

In contrast, our approaches to developing sensor control policies are farsighted. In other words, the policy accounts for dynamics over a time horizon that is longer than it takes to complete the next control action. Two reasons why this is beneficial are that it will appropriately value the placement of a control in time and it will appropriately value actions that do not have immediate benefits. An example of the first case is a scenario in which the target states are changing in time so that the value of different control actions is changing in time. A farsighted policy will account for this and postpone certain sensor actions to opportunistic times in the future even if there are some immediate benefits. Such a policy is especially beneficial if alternative control actions require different amounts of time to complete. In this case, the evolution of the target states over the period required for the control action to complete will be significantly different for the alternate controls. Selecting the best sequence of controls requires accounting for these dynamics. An example of the second benefit often occurs when rewards reflect threshold objectives. For example, one may want to achieve a particular level of accuracy in the state estimate. The reward function may then be modeled as taking the value 0 if the objective is not met and 1 if it is met. As a result, there may be no immediate benefits to taking a particular sensor action. A farsighted policy, however, would value a particular sensor action accounting for the potential for achieving one's objective in the future.

Our approach to developing farsighted policies that are computable was to decompose the problem into subproblems. As mentioned previously, the individual target states are often modeled as independent. Thus, a natural approach to decomposing the computation of the policy is across targets. The problem of computing the sensor control policy is thus split into $n$ distinct

subproblems where *n* is the number of targets. Each of these subproblems is formulated in terms of the information particular to an individual target. This approach to decomposing the problem has the potential to increase the efficiency of computing the sensor control policy by solving the *n* simpler subproblems rather than performing the computation on the single aggregate problem that includes the collection of information across all of the targets. This structure will not always be optimal. However, algorithms having this structure may be optimal in some situations and near-optimal in others. Moreover, some algorithms having the structure may not yield a feasible sequence of sensor controls but may be used to generate a bound on the optimal value of a sensor management problem. We have specifically investigated approaches to computing the following three quantities: lower bounds on the optimal sensor management performance; good, suboptimal strategies for sensor management; and optimal sensor management strategies.

1. **Lower bounds on optimal sensor management performance**. We were interested in deriving bounds for the problem of dynamic adaptive scheduling of multi-mode sensor resources when classifying multiple unknown objects. Sensor schedules are adapted based on the observed data, and the objective is specified in terms of a terminal cost. We were interested in comparing the performance of farsighted and myopic strategies on such problems. Lower bounds on performance allow us to determine how close to optimal candidate strategies may be. We were particularly interested in deriving bounds based on relaxations for which the optimal policy can be expressed as a mixture of farsighted sensor policies, each of which is local in the sense that it only depends on the information concerning an individual target.

2. **Good, suboptimal strategies for sensor management**. We were also investigating techniques for developing good, computable, suboptimal strategies for sensor management problems with no known computable, optimal strategies. A particular focus had been on developing farsighted strategies and determining the benefits such strategies may have over myopic strategies. We have been specifically interested in farsighted strategies whose computation involves a decomposition into single sensor problems.

3. **Optimal sensor management strategies**. Another topic of study was the development of techniques for deriving optimal sensor management strategies. As mentioned previously, computing optimal sensor management strategies in general is often intractable. For special cases, there exist techniques for deriving an optimal solution that can be computed efficiently, often as a result of a decomposition of the problem into a set of single target problems. Being able to compute optimal solutions is useful for many reasons including the following two. First, the quality of performance bounds or suboptimal strategies can be evaluated by comparing the performance predicted by the bounds or resulting from the suboptimal strategies on a special sensor management problem for which one can compute the optimal solution. Second, the optimal solution to a special sensor management problem can be incorporated as a component of a suboptimal solution to a more general sensor management problem. Various techniques exist currently for deriving optimal solutions to sensor management problems. For example, a particular class of problems for which techniques exist is the class of multi-armed bandit problems. However, existing techniques do not apply to the sensor management problems of interest in this program. We were investigating novel techniques for computing optimal sensor management strategies that could be used to verify the quality of bounds or of suboptimal sensor management strategies developed for this program.

We analyzed these approaches in the context of ISR scenarios with air-to-ground sensors. Sensors in such a scenario are being used to detect, track, and classify ground targets. The sensors include agile airborne radars, which have a constrained field of view but can be instantaneously steered to observe a specific area on the ground within a wide field of regard. In addition, the sensors may be able to operate in different modes. Each mode may have distinct characteristics and be suitable for observing specific activities.

## 3.2 ACCOMPLISHMENTS

Our accomplishments during the program have included those in the following list. Details on each of these are provided in the subsequent sections and appendices.

- **Developed a bound on optimal sensor management performance.** The bound is derived by relaxing the problem. In particular, almost sure constraints in the problem are relaxed to expected-use constraints. *For the relaxed problem, we have proved that the optimal sensor control policy can be expressed as a mixture of local sensor policies.* In this setting, a local sensor policy is one which is only a function of a single target's state. The value of the optimal control can thus be computed by solving sub problems associated with each target. We have verified that the bound is tight for a special case for which we derived the optimal strategy. The bound has been used to evaluate the performance of farsighted and myopic strategies to manage a sensor to classify targets.
- **Developed and analyzed farsighted sensor management strategies for controlling a multimode sensor.** In particular, we developed two types of farsighted algorithms for managing a sensor. The algorithms control where the sensor points as well as the mode to use. The sensor modes are assumed to require different amounts of time and be suitable for observing targets in particular states. We also identified a myopic strategy for controlling this type of sensor. All three techniques were evaluated in a simulation of an ISR scenario in which a multimode radar is used to track targets as they start and stop. The results indicate that the farsighted algorithms perform better than the myopic approach.
- **Derived a novel set of sufficient conditions for optimality in Markov decision problems.** The conditions apply to finite-horizon Markov decision problems with a terminal reward. We applied the conditions to verify the optimality of strategies we developed for two different sensor management problems. The first is a class of symmetric binary classification problems. Specifically, a single sensor is being managed to collect discriminatory data to classify targets being one of two types. The second is a class of search problems. The first type of problem is also one for which the performance bounds apply. Thus, the conditions derived for optimality enabled us to investigate the quality of performance bounds by comparing the bound to the optimal performance for a special case.
- **Presented an overview paper on sensor management at the IEEE Automatic Control Conference**. The paper provides an overview of the problem of managing sensor resources in a closed-loop sensor fusion system. We formulated the problem in a stochastic dynamic programming framework. In so doing, we exposed structure in the problem resulting from target dynamics being independent and discussed how this can be exploited in solution strategies. We illustrated situations in which we believe such sensor

management techniques are especially beneficial with two examples. The focus of both examples was on air-to-ground tracking.

- **Submitted a paper on farsighted sensor management strategies for move/stop tracking to the International Conference on Information Fusion.** We considered the sensor management problem arising in using a multi-mode sensor to track moving and stopped targets. The sensor management problem is to determine what measurements to take in time so as to optimize the utility of the collected data. Finding the best sequence of measurements is a hard combinatorial problem due to many factors, including the large number of possible sensor actions and the complexity of the dynamics. The complexity of the dynamics is due in part to the sensor dwell-time depending on the sensor mode, targets randomly starting and stopping, and the uncertainty in the sensor detection process. For such a sensor management problem, we proposed a novel, computationally efficient, farsighted algorithm based on an approximate dynamic programming methodology. The algorithm's complexity is polynomial in the number of targets. We evaluated this algorithm against a myopic algorithm optimizing an information-theoretic scoring criterion. Our simulation results indicate that the farsighted algorithm performs better with respect to the average time the track error is below a specified goal value.
- **Submitted a paper on bounding optimal sensor management performance to the IEEE Conference on Decision and Control.** We considered a network of sensors, each of which has limited sensing resources, which is tasked with collecting noisy classification information on a group of unknown objects. The amount of resources required a given sensor to measure an object depends on the specific sensor-object geometry. Sensors exchange collected information to estimate object identities and coordinate which measurements to collect next. This paper describes a computable lower bound on the classification error that can be achieved by a causal adaptive sensing schedule. This bound is based on a formulation of the adaptive sensing problem as a partially observed stochastic control problem. Expanding the admissible control space of this problem leads to a relaxed problem with simpler decision structure for which the bounds can be computed. The bound computations are illustrated for several examples involving 100 unknown objects, and compared with the Monte Carlo performance of specific adaptive sensor scheduling algorithms. Comparisons with optimal scheduling algorithms for special cases illustrate the tightness of the bounds.

What follows in the next few sections and appendices are details on the accomplishments summarized in Section 3.2. Each of the sections provides a self-contained presentation on one of the accomplishments.

# 4 FARSIGHTED ALGORITHMS FOR CONTROLLING SENSOR MODES

## 4.1 INTRODUCTION

Here, we focus on investigating the advantages of applying farsighted policies to sensor resource management (SRM) problems. In particular, we are interested in determining if there are SRM problems where using farsighted policies is beneficial. We want to explore and characterize such problems, as well as find out what kind of benefits we may expect.

An SRM problem is typically characterized by a set of targets of interest, a specific mission objective, and a set of available sensors. The goal of the sensor manager is to allocate the sensor resources among the targets in time to support the mission success. The sensor resources have to be allocated in the presence of uncertainty associated with obtaining a measurement (e.g., those uncertainties resulting from the sensor detection process).

Our conjecture is that a farsighted approach is better than a myopic approach for the SRM problem where the sensor dwell-time required to complete a sensor task is different for different tasks. A myopic approach selects a sensor action based on the current information only. Thus, this approach is oblivious to the time required to complete the selected action and of future benefits resulting from the action. On the other hand, a farsighted approach accounts for the action's benefits as well as the time it takes to receive benefits. We believe that a farsighted approach having the ability to anticipate future consequences resulting from the actions taken now will yield sensor schedules that manage the resources better and support the mission success better than a myopic approach.

To support our conjecture, we consider analyzing a sensor management problem arising in move/stop tracking with multimode radar. In particular, we are interested in tracking ground targets through their motion state transitions with a radar sensor having two modes: an MTI mode for detecting moving targets only and an FTI mode for detecting stopped targets only. These modes have dwell durations that differ by an order of the magnitude; an FTI dwell is about 100 times longer than an MTI dwell, which adds to the problem complexity. The goal is to manage sensor resources to provide continuous tracking of the targets.

For this problem, we compare a myopic entropy-based SRM algorithm to farsighted SRM algorithms. The farsighted SRM algorithms generate sensor actions by evaluating an objective function parameterized by predictions of target state. The objective function is constructed to value the future benefits of a measurement obtained now. The algorithms also estimate the probability of a target sitting or moving from past reports so as to evaluate the expected benefits of MTI and FTI sensor modes. The expected benefit evaluation accounts for the dwell-time of each mode. The algorithms address the combinatorial complexity of the problem through the use of a rollout approach, which is described in the next section.

## 4.2 BASIC PRINCIPLES OF ROLLOUT ALGORITHMS

A rollout algorithm is a dynamic programming approach that evaluates an action by estimating near and far-future benefits resulting from the current state and the action choice. The near-

future benefits are computed by predicting the action's consequences over the look-ahead planning stages. The far-future benefits are the benefits accumulated after the look-ahead period. In the rollout approach, the far-future benefits are computed as the benefits resulting from applying a fixed policy. Figure 2 illustrates the rollout approach.

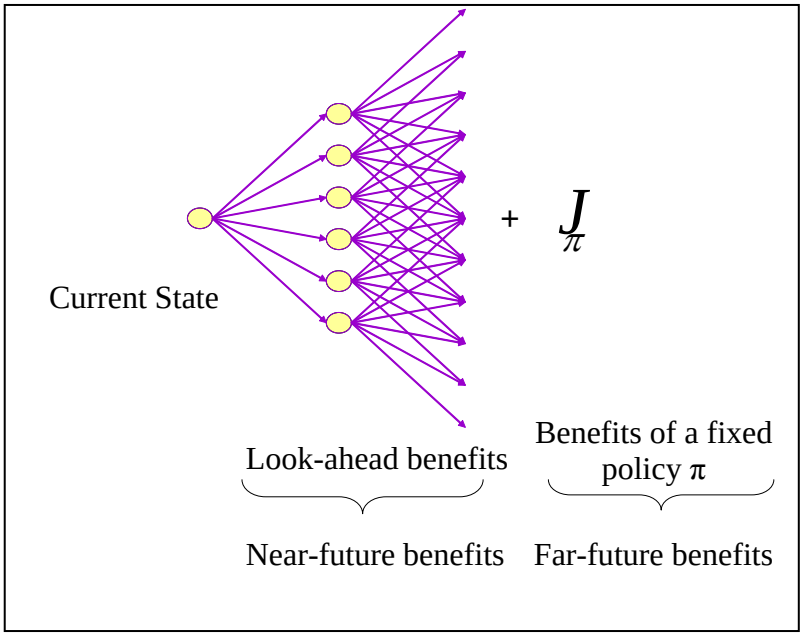


**Figure 2: A rollout approach to evaluate near and far future benefits of an action taken at the current state**

Now we introduce some notation and formally describe the rollout approach, starting with the optimality principle of the dynamic programming theory. In particular, for a discrete time system, the optimality principle states that every optimal policy $\pi^*$ satisfies the following relation:

$$\pi^*(S) = \arg\max_{u \in U(S)} \left\{ \bar{R}(S,u) + \alpha \underset{w}{E}\left[ J^*(f(S,u,w)) \right] \right\}, \quad \text{for all states } S, \tag{1}$$

where

- $S$ is state of the system
- $U(S)$ is the set of controls available in state $S$
- $\bar{R}(S,u)$ is the instantaneous reward received at state $S$ for selecting control $u$
- $\alpha$ is a discount factor satisfying $\alpha \in (0,1)$
- $w$ is a random outcome of control $u$
- $J^*$ is the optimal reward (the reward collected under policy $\pi^*$)
- $f(S,u,w)$ is the function specifying the new state to which the system transitions from state $S$ under control $u$ and the control outcome $w$.

The rollout approach generates a policy $\tilde{\pi}$ by replacing the term $J^*(f(S,u,w))$ in Equation (1) with a reward $J_\pi$ collected from state $f(S,u,w)$ under some policy $\pi$. The resulting policy $\tilde{\pi}$ is a one-step look-ahead policy satisfying the following relation

$$\tilde{\pi}(S) = \arg\max_{u \in U(S)} \left\{ \bar{R}(S,u) + \alpha \underset{w}{E}\left[ J_\pi(f(S,u,w)) \right] \right\}, \quad \text{for all states } S, \tag{2}$$

where $\pi$ is some policy whose reward $J_\pi$ can be efficiently computed. This relation defines a rollout approach that we use to solve our SRM problem. We consider two different implementations, one maximizing precision and one minimizing error as discussed in the following section.

## 4.3 SRM ALGORITHMS FOR MOVE-STOP TRACKING

### 4.3.1 Precision Maximizing Algorithm

Here we formulate the SRM problem for move-stop tracking as a dynamic programming problem, and we present a farsighted SRM algorithm for solving it. Initial development of this algorithm was performed under a SBIR program [4]. That work assumes the dwell times of all modes are the same duration. The extension considered here addresses issues associated with the modes having different durations.

#### 4.3.1.1 Formulation

We model the SRM problem for move-stop tracking as an infinite-horizon, continuous-time stochastic dynamic programming problem. The system to be controlled is the tracker. The state in the dynamic program consists of the target track states. Here, a control choice is specified by a target at which to look and the sensor mode to use. At any time and any state, the available control options are to look at any of the targets currently in track and to use one of the two sensor modes. A sensor measurement is a random outcome of the control choice and affects the future evolution of the tracker state.

For a reward, we chose a function that rewards states with sufficiently high precision (i.e., small error). In particular, the total expected reward accumulated has the following form

$$E\left[ \int_0^\infty e^{-\gamma \cdot t} \sum_{i=1}^{n} V_i \cdot R_i(S_i(t)) dt \right], \tag{3}$$

where

- $\gamma$ is a discount factor specifying the rate at which the future controls contribute to the total reward,
- $n$ is the number of tracks currently in the tracker
- $V_i$ is a priority value of target $i$
- $R_i(\cdot)$ is the instantaneous reward (discussed below)

- $S_i(t)$ is the state of track $i$ (the estimated error variance of track $i$ at time $t$)
- $u(t)$ is the control selected at tracker state $(S_1(t),\ldots,S_n(t))$.

The reward $R_i$ is given by

$$R_i(S_i) = \begin{cases} 1 & \text{if } S_i \le G_i, \\ 0 & \text{if } S_i > G_i, \end{cases} \tag{4}$$

where $G_i$ is the goal state for track $i$ (desired error variance for track $i$).

#### 4.3.1.2 Precision Maximizing SRM

The sensor management problem is to find a sequence of controls maximizing the total expected reward shown in Equation (3). In what follows, we describe an SRM algorithm that generates an approximate solution. The algorithm relies on two different approximations, model approximation and optimal-policy approximation. The first of type of approximation involves using a prediction model to approximate the track states. The second type of approximation involves use of a near-optimal policy instead of the optimal scheduling policy for the approximate model of the tracking system. We subsequently discuss these two approximations in detail.

**Prediction Model**

The architecture of the SRM system is illustrated in Figure 3. The SRM evaluates the sensor actions, in terms of the objective function, Equation (3), by measuring current and future benefits resulting from an action selected at the current time. The future benefits of an action are computed using the SRM prediction model, which predicts the future target behavior.

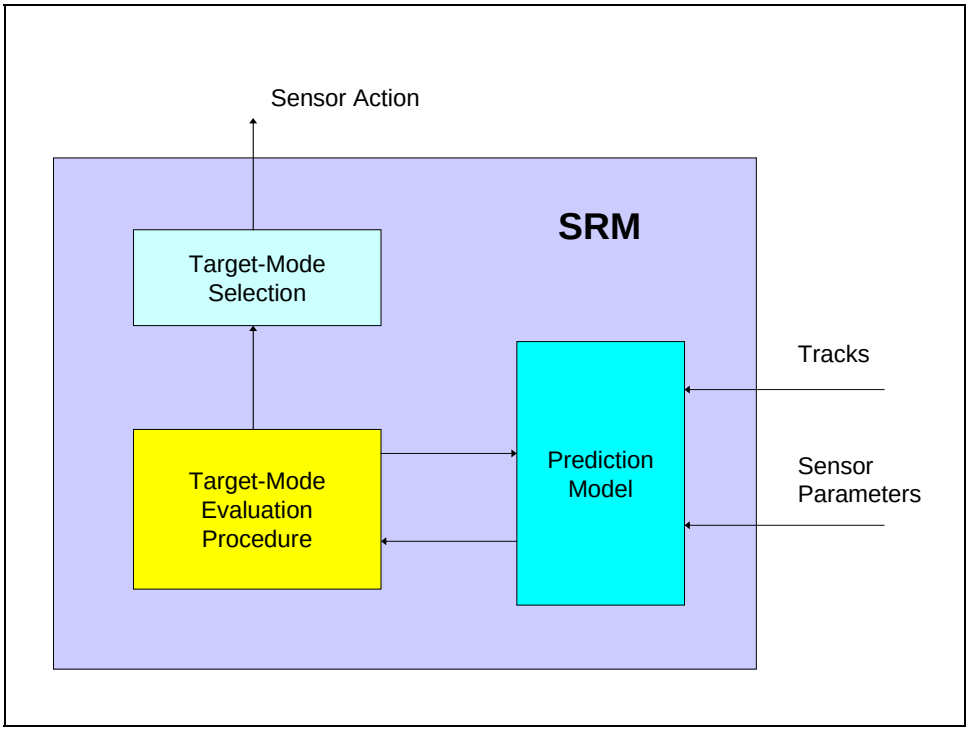


**Figure 3: SRM Architecture**

From the SRM point of view, a target is characterized by a collection of attributes including target mode probability distribution, and target kinematic state, consisting of position error covariance. To accommodate efficient computation of the expected policy-reward, our sensor resource manager uses a tracker predictive model approximating the tracker. This model is based on the following:

*Assumption*:

1. Each target is either moving or is stopped, but the target motion state is unknown.

2. A target track is dropped if the target is not detected.

Assumption 1 is realistic for cases where the changes in target motion take longer than planning and executing a sensor action. Assumption 2 is more conservative than necessary (a target track may continue even with one or more missed detections). However, the resulting model is useful for planning purposes. Furthermore, these assumptions restrict the branching of the control-outcome space of any policy. This allows us to evaluate our farsighted policy without using costly Monte Carlo simulations.

Under this assumption, the probability distribution of the outcomes of any finite sequence of control actions can be computed. We exploit this in our subsequent algorithm development.

The outcome of a single sensor action is either a detection or a missed detection, which results in updating the target states either with or without a measurement. For each of these outcomes, we give the target mode and kinematic state update equations.

*Update with a Measurement*

In this case, the target state is predicted to the next update time and updated with a measurement. The state prediction equation is

$$S_i(t+T) = S_i(t) + T\left(p_{i1}(t)\cdot Q_{i1} + p_{i2}(t)\cdot Q_{i2}\right), \tag{5}$$

where

- $T$ is the time increment,
- $S_i(t)$ is the kinematic state of target $i$ at the current time $t$,
- $S_i(t+T)$ is the predicted state of target $i$ (into the future time $t+T$),
- $(p_{i1}(t),p_{i2}(t))$ is the current probability distribution for the mode of target $i$, with $p_{i1}$ and $p_{i2}$ being the probability of moving and stopped respectively,
- $(Q_{i1},Q_{i2})$ is the process noise covariance vector of the IMM filter, with $Q_{i1}$ and $Q_{i2}$ being the process covariances of the filters modeling respectively moving and stopped modes for target $i$.

The target is detected at time $t+T$ using sensor mode $j$ and the target state is updated as follows:

$$S_i(t+T) = \frac{S_i(t+T)\, r_{ij}}{S_i(t+T) + r_{ij}}, \tag{6}$$

where $r_{ij}$ is the measurement noise covariance for target $i$ and sensor mode $j$.

The target mode probability distribution at the update time is

$$p_{ij}(t+T) = 1, \quad \text{and} \quad p_{im}(t+T) = 0 \quad \text{for modes } m \neq j. \tag{7}$$

*Update without a Measurement*

In this case, the target state is predicted to the next update time according to Equation (5). Since the target is unobserved, the target mode probability distribution does not change according to Assumption 1.

**Algorithm**

As a first step toward maximizing the objective function in Equation (3), we replace the continuous-time optimization problem with a discrete-time approximation. In particular, we discretize the time by letting

$$t_k = k\delta \qquad \text{for} \quad \delta > 0 \quad \text{and} \quad k = 0,1,2,\ldots . \tag{8}$$

We then approximate the continuous-time objective function in Equation (3) with a piece-wise constant function resulting in the following discrete-time objective function

$$\sum_{k=0}^{\infty} \alpha^k \sum_{i=1}^{n} R_i(S_i(t_k), u(t_k)), \tag{9}$$

where $\alpha \in (0,1)$ is a discount factor given by $\alpha = e^{-\gamma\delta}$. For this objective function, we develop a near-optimal policy $\tilde{\pi}$ based on the rollout approach. In particular, for our SRM problem with variable duration dwell times, the rollout relation in Equation (2) takes the following form

$$\tilde{\pi}(S) = \arg\max_{j,m} \left\{ \bar{R}(S, j, m, \Delta_m) + \alpha^{K(m)} \underset{w}{E}\left[ J_\pi(\bar{f}(S, j, m, \Delta_m, w_{jm})) \right] \right\}, \quad \text{for all states } S, \tag{10}$$

where maximization takes place over all target tracks $j$ and sensor modes $m$, and

- $\Delta_m$ is the dwell time for sensor mode $m$
- $\bar{R}(S, j, m, \Delta_m)$ is the reward collected from state $S$ under control $u=(i,m)$ during time interval $\Delta_m$
- $K(m)$ is the dwell time in units of $\delta$ for mode $m$
- $J_\pi$ is the expected reward accumulated under some policy $\pi$
- $\bar{f}(S, j, m, \Delta_m, w_{jm})$ is the state of the tracker at the end of the time interval $\Delta_m$ when the measurement is made,
- $w_{jm}$ is a random variable taking values 1 or 0 that indicate detecting and correctly associating the detection of target $j$ for sensor mode $m$.

Since the MTI dwell time is typically less than the FTI dwell time, one can use the MTI dwell time as a unit of time and express the FTI dwell time as a multiple $\tau_{MTI}$ of it. By viewing $\Delta_{MTI}$ as the time unit measure $\delta$, we have the following values for the dwell times $K(m)$ in Equation (10).

$$K(MTI) = 1 \quad \text{and} \quad K(FTI) = \tau_{MTI}. \tag{11}$$

The mode rewards $\bar{R}(S, j, m, \Delta_m)$ for target $j$ have the following form:

$$\bar{R}(S, j, MTI, \Delta_{MTI}) = \sum_{i=1}^{n} R_i(S_i), \tag{12}$$

$$\bar{R}(S, j, FTI, \Delta_{FTI}) = \sum_{t=0}^{K(FTI)-1} \alpha^t \sum_{i=1}^{n} R_i(S_i(t)), \tag{13}$$

where $S=(S_1,\ldots,S_n)$ is the current tracker state and $S(t)$ is its state $t$ units later [$S(0) = S$], $S_i$ is the current state of track $i$ and $S_i(t)$ is its state $t$ units later, while the reward $R_i$ is given by Equation (4).

We next describe the evaluation of the term

$$\underset{w}{E}\left[ J_\pi(\bar{f}(S, j, m, \Delta_m, w_{ij})) \right] \tag{14}$$

in the right hand side of the rollout Equation (10). The expectation is with respect to the two possible outcomes $w_{jm}=1$ and $w_{jm}=0$ indicating a detection of target $j$ with sensor mode $m$. Hence, we have

$$\begin{aligned} \underset{w}{E}\left[ J_\pi(\bar{f}(S, j, m, \Delta_m, w_{jm})) \right] &= J_\pi(\bar{f}(S, j, m, \Delta_m, 1)) Prob\{w_{jm} = 1\} \\ &\quad + J_\pi(\bar{f}(S, j, m, \Delta_m, 0)) Prob\{w_{jm} = 0\}, \end{aligned} \tag{15}$$

The probability of outcomes $w_{jm}=1$ and $w_{jm}=0$ are given by

$$Prob\{w_{jm}=1\}=p_{im}\beta_{im}, \qquad Prob\{w_{jm}=0\}=1-p_{im}\beta_{im}, \tag{16}$$

where

- $p_{jm}$ is the probability that the target mode is $m$ (matching the sensor mode)
- $\beta_{jm}$ is the detection probability of sensor mode $m$ for target $j$.

The target mode probabilities $p_{jm}$ are computed using the mode likelihoods generated by the IMM filter.

We now discuss the choice of policy $\pi$. Motivated by the desire to have a good policy whose expected reward can be computed for any initial state, *we consider a policy $\pi$ having the following properties*:

1. A target is observed with either MTI or FTI mode at all times.

2. Initially, the targets are sorted in a list according to some criterion. Then, these targets are observed according to the list as follows: each target is observed until either its track error decreases below the desired value or its track is dropped. If the track error is decreased below the desired value, the target is revisited at the rate that keeps its track error below the desired value.

We assume that the sensor can revisit the targets with the rates that keep the track errors below the desired values.

Under the policy $\pi$, it is assumed that the sensor uses one and the same mode when observing a target. The sensor mode $m(i)$ to be used for observing target $i$ is determined as the most likely target mode in the current target mode probability distribution $p_i = (p_{i1}, p_{i2})$, i.e.,

$$m(i)=\arg\max\{p_{i1}, p_{i2}\}. \tag{17}$$

Given the current tracker state $\overline{S}=(\overline{S}_1,...,\overline{S}_n)$, policy $\pi$ sorts the tracks according to their vicinity to the goal state, i.e., the tracks are sorted according to the values $\{\overline{S}_i / G_i \mid i=1,2,...,n\}$. The order is motivated by that generated according to an index rule policy such as that discussed in [4]. The targets are then considered in that order, and to each target track the following rule is applied:

If the target track state $\overline{S}_i$ is outside the track accuracy goal region $\{s \mid s \le G_i\}$, the target is consecutively observed, with the sensor mode matching the target mode, until the time its state $\overline{S}_i(t)$ enters the track accuracy goal region. After that time, the target is periodically revisited with the smallest revisit rate that guarantees the state will remain within the goal region.

It is computationally prohibitive to exactly evaluate the policy reward $J_\pi$ due to unpredictable target mode changes in the future and due to the explosion of possibilities of observation outcomes over a long period of time. We approximately evaluate the policy reward $J_\pi$ using the predictive model for the evolution of the target mode probability distribution and kinematic state, as described earlier.

The policy reward $J_\pi$ has the following form

$$J_{\pi}(\bar{S}) = \sum_{i=1}^{n} J_i(\bar{S}) . \tag{18}$$

For notational convenience assume that the order of the targets is 1, 2,…,*n* when the targets are sorted according to values $\left\{ \bar{S}_i / G_i \mid i = 1, 2, ..., n \right\}$. Then, for each target, we determine the target mode and the associated mode probability. The mode probabilities are derived from the mode likelihoods generated by the IMM filter, and the target mode is defined as the most likely of the modes.

Suppose that the states of the first $\bar{\tau}$ targets are within their goal region, i.e.,

$$S_i \le G_i \qquad \text{for } i = 1, 2, \ldots, \bar{\tau} . \tag{19}$$

For these targets, we have

$$J_i(\bar{S}_i) = L_i, \qquad i = 1, 2, \ldots, \bar{\tau} . \tag{20}$$

where $L_i$ is the long-term reward accumulated during revisits (to be discussed shortly).

Consider now the targets $\bar{\tau}+1,\ldots,n$, which are outside their corresponding goal regions. Suppose that $T_1$ is the observation time required for state $\bar{S}_{\bar{\tau}+1}$ to enter the goal region $\left\{ s \mid s \le G_{\bar{\tau}+1} \right\}$. We then have

$$J_{\bar{\tau}+1}(\bar{S}_{\bar{\tau}+1}) = \left( \alpha \, \beta_{\bar{\tau}+1, m(\bar{\tau}+1)} \right)^{T_1} \cdot L_{\bar{\tau}+1} . \tag{21}$$

where $m(\bar{\tau}+1)$ is the sensor mode that is used for observing target $\bar{\tau}+1$. While observing target $\bar{\tau}+1$, the states of the remaining targets $\bar{\tau}+2,\ldots,n$ have evolved to $\bar{S}_{\bar{\tau}+2}(T_1), ..., \bar{S}_n(T_1)$. Suppose now that $T_2$ is the observation time required for state $\bar{S}_{\bar{\tau}+2}(T_1)$ to enter the goal region $\left\{ s \mid s \le G_{\bar{\tau}+2} \right\}$. Then, we have

$$J_{\bar{\tau}+2}(\bar{S}) = \left( \alpha \, \beta_{\bar{\tau}+1, m(\bar{\tau}+2)} \right)^{T_1+T_2} \cdot L_{\bar{\tau}+2} . \tag{22}$$

Continuing in this manner, we can see that

$$J_{\bar{\tau}+j}(\bar{S}) = \left( \alpha \, \beta_{\bar{\tau}+1, m(\bar{\tau}+j)} \right)^{T_1+T_2+\ldots+T_j} \cdot L_{\bar{\tau}+j}, \qquad \text{for } j = 1, \ldots, n-\bar{\tau}, \tag{23}$$

where $T_j$ is the time for variance $\bar{S}_{\bar{\tau}+j}(T_1 + \ldots + T_{j-1})$ to enter the goal region $\{ s \mid s \le G_{\bar{\tau}+j} \}$.

We now discuss the long-term rewards $L_i$ accumulated during periodic revisits of the targets. As we mentioned earlier, once the states of all targets enter their corresponding goal regions, the targets are revisited at a constant rate. Under the assumption that the target-mode probability distribution does not change, once a stopped target state is within its corresponding goal region, it remains there for the rest of the time. Therefore, the stopped targets are not revisited, and the long-term reward $L_i$ associated with a stopped target *i* is computed as follows

$$L_i = \sum_{t=0}^{\infty} \alpha^t V_i = \frac{V_i}{1-\alpha} . \tag{24}$$

We next discuss the long-term reward $L_i$ associated with a moving target *i*. Let *M* be the length of the revisit interval required for keeping the state of target *i* within the goal region. Without

loss of generality we may assume that the sensor revisits the target $i$ at times $t=jM$, $j=0,1\ldots$. Note that, in view of our assumptions, the revisit process continues for as long as the target is detected, and the process stops when the target is lost as a result of a missed detection. Furthermore, the reward $V_i$ is collected while the target is still tracked, and the reward ceases when the target is lost.

During the revisit stage, the target lifetime is a random variable taking value $jM$ with probability $\beta_{i,m(i)}^{j-1}\left(1-\beta_{i,m(i)}^{j-1}\right)$ for $j=1,2,\ldots$ . Let $\rho$ be the reward accumulated between any two consecutive revisits. When the lifetime takes value $jM$, the lifetime reward $Rew(jM)$ is given by

$$Rew(jM)=\rho+\alpha^M\rho+\ldots+\alpha^{(j-1)M}\rho=\rho\frac{1-\alpha^{jM}}{1-\alpha^M}. \tag{25}$$

The long-term reward $L_i$ is equal to the expected lifetime reward during the revisit period, and it can be seen that

$$L_i=\frac{\rho}{1-\alpha^M\beta_{i,m(i)}}. \tag{26}$$

Since $\rho$ is the reward accumulated during the subsequent revisits, we have

$$\rho=V_i\left(1+\alpha+\ldots+\alpha^{M-1}\right)=V_i\frac{1-\alpha^M}{1-\alpha}. \tag{27}$$

By substituting this $\rho$ in Equation 26, we see that the long-term reward for a moving target $i$ is given by

$$L_i=V_i\frac{1-\alpha^M}{\left(1-\alpha\right)\left(1-\alpha^M\beta_{i,m(i)}\right)}. \tag{28}$$

We note, here, that the preceding precision maximizing SRM algorithm extends to the multidimensional case by replacing the variance $S_i$ with the trace $Tr(S_i)$ of the covariance $S_i$.

### 4.3.2 Error Minimizing Algorithm

Here we present an error minimizing SRM algorithm as applied to move-stop tracking. In the dynamic programming formulation of the SRM problem for move-stop tracking (Section 4.3.1.1), we use an instantaneous cost instead of a reward. In particular, a target incurs a nonzero cost if the target error variance exceeds a specified error variance goal. The target cost is instantaneous, and it is proportional to the difference between the target error and the error goal. Formally, the instantaneous cost at time $t$ for target $i$ is given by

$$C_i(S_i(t)) = V_i \cdot \max\{S_i(t)\text{-}G_i, 0\}, \tag{29}$$

where $S_i(t)$ is the target error variance, $V_i$ is the target priority, and $G_i$ is the error goal for target $i$. The cost of the composite target state $S(t)=(S_1(t),\ldots,S_n(t))$ is additive over the targets, i.e.,

$$C\left(S(t)\right)=\sum_{i=1}^{n}C_i\left(S_i(t)\right). \tag{30}$$

#### 4.3.2.1 Error Minimizing SRM

For a continuous-time system the rollout relation in Equation (2) takes the following form

$$\tilde{\pi}(S) = \arg\min_{u \in U(S)} \left\{ \int_0^{t_u} C\left(S(\tau)\right) e^{-\gamma\tau} d\tau \; + \; e^{-\gamma t_u} \underset{w}{E}\left[ J_\pi( f(S(t_u), u, w)) \right] \right\}, \quad \text{for all states } S, \tag{31}$$

where

- $S$ is the state of targets at the current time $t$=0, i.e., $S$=$S$(0)
- $\pi$ is some fixed policy
- $U(S)$ is the set of controls available in state $S$
- $t_u$ is the latency time associated with control $u$ (for a sensor, $t_u$ is the dwell-time)
- $\gamma$ is an exponential decay factor satisfying $\gamma$>0
- $w$ is a random outcome of the control $u$
- $f(S,u,w)$ is the function specifying the new state to which the system transitions from state $S$ under control $u$ and the control outcome $w$.

We assume that the control options are target-mode pairs, with the set of candidate modes being MTI, FTI, and "idle" (no target selected). We also assume that the "idle mode" has infinite dwell time, so that once the idle mode is selected no other mode can ever be used in the future. We consider a rollout where $\pi$ is the "idle"-policy, i.e., the policy selects the "idle" mode at any state. For such a policy $\pi$ and the additive cost, the minimization on the right-hand side of Equation (31) reduces to:

$$\min_{u \in U(S)} \sum_{i=1}^{n} \underbrace{\left\{ \underbrace{\int_0^{t_u} C_i(S_i(t)) e^{-\gamma t} dt}_{\text{transitional cost}} + \underbrace{e^{-\gamma t_u} \cdot \underset{w}{E} \int_0^{\infty} C_i\left( f(S_i(t_u), u, w)\right) e^{-\gamma t} dt}_{\text{future cost}} \right\}}_{\text{target cost}}, \tag{32}$$

where the set of controls $U(S)$ is the same for all states $S$, i.e., $U(S)$=$U$ for all $S$ with

$$U = \left\{ (i,m) \mid i \in \{1,...,n\}, \;\; m \in \{MTI, FTI, idle\} \right\}. \tag{33}$$

The transitional cost is incurred from the current time until the execution time $t_u$ of control $u$. The future cost is incurred for the rest of the time, starting from the time immediately after the control $u$ is executed. Computing each of these costs requires predicting the evolution of target states. We approximate the state evolution of the IMM tracker by using the predictive model described in Section 4.3.1.2. This simple model estimates the target behavior well and simplifies the integral computations in Equation 32.

**Algorithm**

We now focus on the minimization problem given in Equation (32). At first, we evaluate the target costs in Equation (32) for a given control $u$=($j$,$\mu$) with $\mu \in \{MTI, FTI\}$. For target $j$, we consider the cost for the cases when the target is observed and is unobserved. Without lost of generality, we may assume that the current time is $t$=0, so that the current target state is $S_j$(0).

*Cost for Observed Target*

During the observation, the target state evolves as follows:

$$S_j(t) = S_j(0) + t \cdot \left( p_{j1} Q_{j1} + p_{j2} Q_{j2} \right), \qquad \text{for} \quad t \in [0, t_u), \tag{34}$$

where $p_{j,1}$ and $p_{j,2}$ are the probabilities for target motion modes with 1 corresponding to moving target and 2 corresponding to stopped target. From the preceding relation and the definition of the cost [cf. Equation (29)], it follows that

$$C_j\left(S_j(t)\right) = V_j \cdot \max\left\{ S_j(0) - G_j + t \cdot \left( p_{j1} Q_{j1} + p_{j2} Q_{j2} \right),\ 0 \right\}, \qquad \text{for} \quad t \in [0, t_u). \tag{35}$$

Define

$$T_j = \max\left\{ \frac{G_j - S_j(0)}{p_{j1} Q_{j1} + p_{j2} Q_{j2}},\ 0 \right\}, \tag{36}$$

and note that $T_j$ is the time the target variance $S_j(t)$ exceeds the specified variance bound $G_j$. We refer to $T_j$ as crossing time from state $S_j(0)$. We note that this time depends on the initial variance $S_j(0)$, i.e., $T_j = T_j(S_j(0))$, and it may be infinite.

It can be seen that the transitional cost has the following form

$$\begin{aligned} TranCost_j(S_j(0)) = {} & \frac{V_j}{\gamma} \max\left\{ S_j(0) - G_j,\ 0 \right\} \cdot \left( 1 - e^{-\gamma t_u} \right) \\ & + \frac{V_j \cdot \left( p_{j1} Q_{j1} + p_{j2} Q_{j2} \right)}{\gamma^2} \cdot \left[ \left( \gamma\theta + 1 \right) e^{-\gamma\theta} - \left( \gamma t_u + 1 \right) e^{-\gamma t_u} \right], \end{aligned} \tag{37}$$

where

$$\theta = \min\left\{ t_u,\ T_j \right\}.$$

We now focus on the expected future cost. Let $S_j(t_u,w)$ denote the target state immediately after the observation time $t_u$, where $w$=0 or $w$=1 indicates that the state $S_j(t_u,w)$ results from the update with or without a measurement, respectively. Since the target-mode probability distribution does not change when the target is unobserved (cf. Assumption 1), the future cost from state $S_j(t_u,w)$ for $w$=0 is given by:

$$FutureCost_j(S_j(t_u, 0)) = V_j \cdot \int_{t_u}^{\infty} \max\left\{ S_j(t_u, 0) - G_j + t\left( p_{j1} Q_{j1} + p_{j2} Q_{j2} \right),\ \ 0 \right\} e^{-\gamma t} dt. \tag{38}$$

However, we have $p_{j\mu}$=1 for the observed target. Therefore, the future cost from state $S_j(t_u,w)$ for $w$=1 is given by:

$$FutureCost_j(S_j(t_u, 1)) = V_j \cdot \int_{t_u}^{\infty} \max\left\{ S_j(t_u, 0) - G_j + t p_{j\mu} Q_{j\mu},\ \ 0 \right\} e^{-\gamma t} dt. \tag{39}$$

Let $T_j(w)$ be the crossing time for target $j$ starting from state $S_j(t_u,w)$, i.e.,

$$T_j(0) = \max\left\{\frac{G_j - S_j(t_j,0)}{p_{j1}Q_{j1} + p_{j2}Q_{j2}},\ 0\right\}, \tag{40}$$

$$T_j(1) = \max\left\{\frac{G_j - S_j(t_j,0)}{p_{j\mu}Q_{j\mu}},\ 0\right\}. \tag{41}$$

Then, it can be seen that the future cost accumulated from state $S_j(t_u,w)$ is given by

$$FutureCost_j(S_j(t_u,w)) = \frac{V_j}{\gamma}\max\left\{S_j(t_u,w) - G_j,\ 0\right\}e^{-\gamma t_u} \ + \ \frac{V_j \cdot Q_j(w)}{\gamma^2}\left(\gamma\cdot\tau(w)+1\right)e^{-\gamma\tau}, \tag{42}$$

where

$$Q_j(w) = \begin{cases} p_{j1}Q_{j1} + p_{j2}Q_{j2} & \text{for } w=0, \\ p_{j\mu}Q_{j\mu} & \text{for } w=1, \end{cases} \tag{43}$$

$$\tau(w) = \max\left\{t_u,\ T_j(w)\right\} \qquad \text{for } w = 0,1. \tag{44}$$

The expected future cost is a weighted sum of the future costs corresponding to outcomes $w$=1 and $w$=0. Specifically, it is given by

$$ExpFutureCost_j(S_j(t_u)) = \beta_{j\mu}p_{j\mu}FutureCost_j(S_j(t_u,1)) + \left(1-\beta_{j\mu}p_{j\mu}\right)FutureCost_j(S_j(t_j,0)), \tag{45}$$

where $\beta_{j\mu}$ is the probability of detecting target $j$ with sensor mode $\mu$.

Then, the total cost for target $j$ is

$$\begin{aligned} TotalCost_j(S_j(0)) = {} & TranCost_j(S_j(0)) + FutureCost_j(S_j(t_u,0)) \\ & + \beta_{j\mu}p_{j\mu}\left[FutureCost_j(S_j(t_u,1)) - FutureCost_j(S_j(t_u,0))\right]. \end{aligned} \tag{46}$$

The first two terms represent the cost associated with the event of not observing the target. The last term represents the expected cost reduction resulting from target detection. Note that the last term is non-positive, i.e.,

$$\beta_{j\mu}p_{j\mu}\left[FutureCost_j(S_j(t_u,1)) - FutureCost_j(S_j(t_u,0))\right] \le 0. \tag{47}$$

*Cost for Unobserved Target*

For a target $i$ with $i \neq j$, there is no uncertainty in the future cost, so that the total cost is given by

$$TotalCost_i(S_i(0)) = TranCost_i(S_i(0)) + FutureCost_i(S_i(t_u,0)), \tag{48}$$

where transient cost $TranCost_i(S_i(0))$ and future cost $FutureCost_i(S_i(t_u,0))$ are given by Equation (37) and Equation (38), respectively.

By summing the total costs of all targets, we obtain the cost associated with the state $S(0)$ and the control choice $u=(j,\mu)$ for $\mu \in \{MTI, FTI\}$. In particular, we have

$$\begin{aligned} Cost(S(0),\, j, \mu) = & \sum_{i=1}^{n} \left[ TranCost_i(S_i(0)) + FutureCost_i(S_i(t_u, 0)) \right] \\ & + \beta_{j\mu} p_{j\mu} \left[ FutureCost_j(S_j(t_j, 1)) - FutureCost_j(S_j(t_j, 0)) \right]. \end{aligned} \tag{49}$$

The cost associated with the control option $u = (j, idle)$ is

$$Cost(S(0),\, j, idle) = \sum_{i=1}^{n} \left[ TranCost_i(S_i(0)) + FutureCost_i(S_i(t_u, 0)) \right]. \tag{50}$$

In view of Equation (47), the cost of control $u=(j,\mu)$ for $\mu \in \{MTI, FTI\}$ is smaller than the cost of control $u = (j, idle)$ a fixed target $j$, i.e.,

$$\begin{aligned} Cost(S(0),\, j, \mu) = & \sum_{i=1}^{n} \left[ TranCost_i(S_i(0)) + FutureCost_i(S_i(t_u, 0)) \right] \\ & + \beta_{j\mu} p_{j\mu} \left[ FutureCost_j(S_j(t_j, 1)) - FutureCost_j(S_j(t_j, 0)) \right] \\ \leq & \sum_{i=1}^{n} \left[ TranCost_i(S_i(0)) + FutureCost_i(S_i(t_u, 0)) \right] \\ = & \, Cost(S(0),\, j, idle). \end{aligned} \tag{51}$$

Hence, the minimization in Equation (32) reduces to

$$\min_{\substack{j \in \{1,\ldots,n\} \\ \mu \in \{MTI, FTI\}}} Cost(S(0),\, j, \mu), \tag{52}$$

with the control cost $Cost(S(0),j,\mu)$ as given in Equation (49).

The preceding error minimizing SRM algorithm extends to the multidimensional case by replacing the variance $S_i$ with the trace $Tr(S_i)$ of the covariance $S_i$.

#### 4.3.2.2 Myopic Entropy-Based SRM Algorithm

Here, we present a myopic sensor management algorithm that serves as a baseline for evaluating the performance of the farsighted algorithms discussed in the preceding sections. We do not consider a myopic approach optimizing the dynamic programming formulation. This is because the myopic property is not well defined for the SRM problems where different control actions have significantly different execution time. In particular, this is the case with the SRM problem for move/stop tracking, where the system state transition time is significantly different for different controls (specifically, for different sensor modes). Thus to anticipate benefits at the possible future states, we have to predict into the future over significantly different time intervals, which is not a property of a myopic approach.

We consider an algorithm that evaluates sensor actions based on the expected decrease in the entropy of the target-track errors per unit of time. The algorithm is myopic since the changes in the entropy are computed only for a single sensor action. Specifically, let the current time be $t=0$ and let the current system state be $S=(S_1,\ldots,S_n)$. The entropy $h_i$ for target $i$ with variance $S_i$ and the mode probabilities $(p_{i1}, p_{i2})$ is given by

$$h_i(S_i) = V_i\left(\frac{1}{2}p_{i1}\log\left[2\pi_c e S_i\right] + \frac{1}{2}p_{i2}\log\left[2\pi_c e S_i\right] - p_{i1}\log p_{i1} - p_{i2}\log p_{i2}\right)$$
$$= \frac{V_i}{2}\log\left[2\pi_c e\right] + \frac{V_i}{2}\log S_i - V_i\left(p_{i1}\log p_{i1} + p_{i2}\log p_{i2}\right), \tag{53}$$

where $V_i$ is the priority of target $i$ and $\pi_c \approx 3.14$ (see [5], Chapter 9). As seen from this relation, the target entropy is measured by the target-track error in log-scale. The entropy $H$ of the system the current time is defined as the sum of the current target entropies $h_i$:

$$H(S) = \sum_{i=1}^{n} h_i(S_i) \qquad \text{where } S = (S_1, ..., S_n). \tag{54}$$

An entropy-based score is associated with each control $u=(j,\mu)$. The score is equal to the expected decrease in the entropy per unit of time:

$$D(S,u) = \frac{\underset{w}{E}\left[H(S(t_u))\right] - H(S)}{t_u}, \tag{55}$$

where $S(t_u)$ is the state to which the system transitions under the control $u$.

At any decision time, the entropy-based SRM algorithm selects a control having the minimum score i.e., the sensor manager solves the following problem

$$\min_{u \in U} D(S,u)$$
$$U = \left\{(j,\mu) \mid j \in \{1,...,n\},\ \mu \in \{MTI, FTI\}\right\}. \tag{56}$$

We now derive the explicit form for $D(S,u)$. Under any control, the predicted target variances at time $t_u$ are

$$S_i^+(t_u) = S_i + t_u \cdot (p_{i1}Q_{i1} + p_{i2}Q_{i2}). \tag{57}$$

Thus, the entropy at time $t_u$ for unobserved target $i$ is given by

$$h_i(S_i^+(t_u)) = \frac{V_i}{2}\log\left[2\pi_c e\right] + \frac{V_i}{2}\log S_i^+(t_u) - V_i\left(p_{i1}\log p_{i1} + p_{i2}\log p_{i2}\right). \tag{58}$$

Under a control $u=(j,\mu)$ and outcome $w=1$ (corresponding to target detection), the updated variance of target $j$ is

$$\hat{S}_j(t_u) = \frac{S_j^+(t_u)\cdot r_{j,\mu}}{S_j^+(t_u) + r_{j,\mu}}. \tag{59}$$

We have $p_{j\mu}=1$ in this case, so that the entropy at time $t_u$ for observed target $j$ is given by

$$h_j(\hat{S}_j(t_u)) = \frac{V_j}{2}\log\left[2\pi_c e\right] + \frac{V_j}{2}\log \hat{S}_j(t_u). \tag{60}$$

For a given control $u=(j,\mu)$, the variances $S_i$ of targets $i$ with $i\neq j$ do not depend on the control outcome $w$, the following holds for the expected entropy at time $t_u$:

$$\begin{aligned}\underset{w}{E}\left[H(S(t_u))\right] &= \sum_{\substack{i=1\\ i\neq j}}^{n} h_i(S_i^+(t_u)) + \beta_{j,\mu}p_{j,\mu}h_j(\hat{S}_j(t_u)) + \left(1-\beta_{j,\mu}p_{j,\mu}\right)h_j(S_j^+(t_u)) \\ &= \sum_{i=1}^{n} h_i(S_i^+(t_u)) + \beta_{j,\mu}p_{j,\mu}\left(h_j(\hat{S}_j(t_u)) - h_j(S_j^+(t_u))\right).\end{aligned} \tag{61}$$

By substituting the expression for the target entropies $h_i$ [cf. Equations (25) and (60)], we obtain

$$\begin{aligned}\underset{w}{E}\left[H(S(t_u))\right] &= \frac{1}{2}\log\left[2\pi_c e\right]\sum_{i=1}^{n}V_i + \frac{1}{2}\sum_{i=1}^{n}V_i\log S_i^+(t_u) - \sum_{i=1}^{n}V_i\left(p_{i1}\log p_{i1} + p_{i2}\log p_{i2}\right) \\ &\quad + \beta_{j,\mu}p_{j,\mu}V_j\left(\frac{1}{2}\log\hat{S}_j(t_u) - \frac{1}{2}\log S_j^+(t_u) + p_{j1}\log p_{j1} + p_{j2}\log p_{j2}\right).\end{aligned} \tag{62}$$

The expected entropy change is given by

$$\begin{aligned}\underset{w}{E}\left[H(S(t_u))\right] - H(S) &= \frac{1}{2}\log\left[2\pi_c e\right]\sum_{i=1}^{n}V_i + \frac{1}{2}\sum_{i=1}^{n}V_i\log S_i^+(t_u) - \sum_{i=1}^{n}V_i\left(p_{i1}\log p_{i1} + p_{i2}\log p_{i2}\right) \\ &\quad + \beta_{j,\mu}p_{j,\mu}V_j\left(\frac{1}{2}\log\hat{S}_j(t_u) - \frac{1}{2}\log S_j^+(t_u) + p_{j1}\log p_{j1} + p_{j2}\log p_{j2}\right) \\ &\quad - \frac{1}{2}\log\left[2\pi_c e\right]\sum_{i=1}^{n}V_i - \frac{1}{2}\sum_{i=1}^{n}V_i\log S_i + \sum_{i=1}^{n}V_i\left(p_{i1}\log p_{i1} + p_{i2}\log p_{i2}\right),\end{aligned} \tag{63}$$

which reduces to

$$\begin{aligned}\underset{w}{E}\left[H(S(t_u))\right] - H(S) &= \frac{1}{2}\sum_{i=1}^{n}V_i\log S_i^+(t_u) - \frac{1}{2}\sum_{i=1}^{n}V_i\log S_i \\ &\quad + \beta_{j,\mu}p_{j,\mu}V_j\left(\frac{1}{2}\log\hat{S}_j(t_u) - \frac{1}{2}\log S_j^+(t_u) + p_{j1}\log p_{j1} + p_{j2}\log p_{j2}\right) \\ &= \frac{1}{2}\sum_{i=1}^{n}V_i\log\frac{S_i^+(t_u)}{S_i} + \beta_{j,\mu}p_{j,\mu}V_j\left(\frac{1}{2}\log\frac{\hat{S}_j(t_u)}{S_j^+(t_u)} + p_{j1}\log p_{j1} + p_{j2}\log p_{j2}\right).\end{aligned} \tag{64}$$

Thus, the myopic entropy-based SRM selects a control $u$ that minimizes the time averaged changes in the entropy:

$$\frac{\underset{w}{E}\left[H(S(t_u))\right] - H(S)}{t_u} = \frac{1}{2t_u}\sum_{i=1}^{n}V_i\log\frac{S_i^+(t_u)}{S_i} + \frac{\beta_{j,\mu}p_{j,\mu}V_j}{t_u}\left(\frac{1}{2}\log\frac{\hat{S}_j(t_u)}{S_j^+(t_u)} + p_{j1}\log p_{j1} + p_{j2}\log p_{j2}\right), \tag{65}$$

over all $u\in U$, where $U=\left\{(j,\mu)\,|\,j\in\{1,...,n\},\ \mu\in\{MTI,FTI\}\right\}$.

For the multidimensional case, the entropy of a target $i$ is given by

$$h_i(S_i) = \frac{NV_i}{2}\log\left[2\pi_c e\right] + \frac{NV_i}{2}\log\left|S_i\right| - V_i\left(p_{i1}\log p_{i1} + p_{i2}\log p_{i2}\right), \tag{66}$$

where $S_i$ is the covariance matrix, $N$ is the size of the matrix $S_i$, and $|S_i|$ is the determinant of $S_i$. In this case, the myopic SRM selects a control $u$ that minimizes (over all $u \in U$) the time averaged changes in the entropy:

$$\frac{\underset{w}{E}\left[H(S(t_u))\right] - H(S)}{t_u} = \frac{1}{2t_u}\sum_{i=1}^{n} NV_i \log\frac{\left|S_i^+(t_u)\right|}{\left|S_i\right|} + \frac{\beta_{j,\mu}p_{j,\mu}V_j}{t_u}\left(\frac{N}{2}\log\frac{\left|\hat{S}_j(t_u)\right|}{\left|S_j^+(t_u)\right|} + p_{j1}\log p_{j1} + p_{j2}\log p_{j2}\right). \tag{67}$$

.

### 4.4 SIMULATION RESULTS

Here, we present our simulation model and the test results obtained for the precision maximizing algorithm, the error minimizing algorithm, and the myopic entropy-based algorithm.

#### 4.4.1 Model Parameters

We start with a detailed discussion of the tracker and sensor parameters, and values identified to be reflective of realistic scenarios.

**Tracker Model Parameters**

In the tracker, the target dynamics are modeled by an interacting multiple model (IMM) filter (see [6]) consisting of two filters: one modeling the kinematics of a "moving" target and the other modeling the kinematics of a "stopped" target. It is assumed that, when moving, targets move along a road (i.e., along a line) with a random velocity normally distributed with specified root mean square value. The target kinematic state consists of the target location estimate and the estimate of error variance. These states are estimated from position reports generated by a single sensor.

Perfect report association is assumed, so that each report is associated with a single target. The tracker model drops the track if the target position error variance exceeds a specified maximum value. A new track is immediately initialized based on the target truth.

The target motion mode, moving or stopped, is modeled according to a discrete-time Markov chain with state dependent transition probabilities, as illustrated in Figure 4.

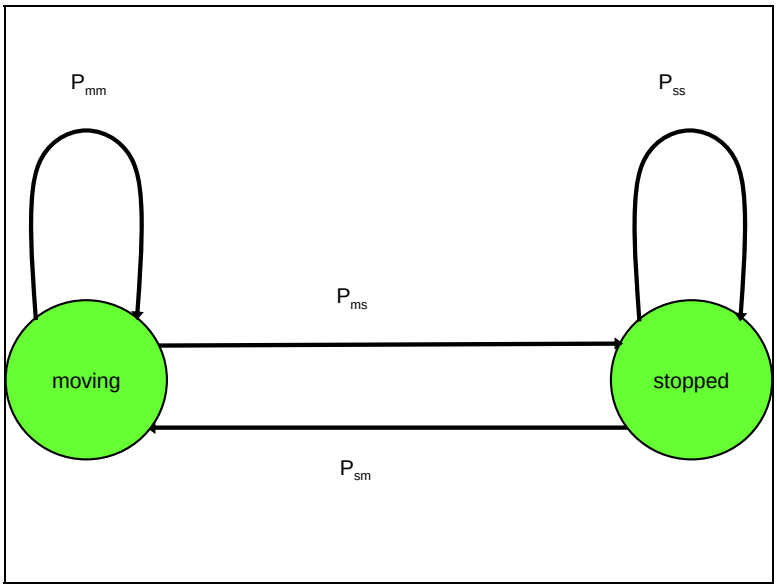


**Figure 4: Discrete-time Markov chain modeling target motion**

At any time, a target can be in one of the two motion states: moving or stopped. The target state transitions occur at times $t_k = k\delta$, $k = 1, 2, \ldots$, where $\delta$ is the time increment. The transition probabilities $P_{ij}$ are state dependent, in particular $P_{ij} = Prob\{\text{next state is } j \mid \text{current state is } i\}$ for $i, j \in \{m, s\}$, where $m$ and $s$ denote moving and stopped respectively.

By specifying the average number of the targets stopped in the steady state in a scenario, we derive the transition probabilities for the Markov chain model that are appropriate for the scenario. In particular, the average number of stopped targets in the steady state is given by

$$\text{average number of targets stopped} = \frac{nP_{ms}}{P_{ms} + P_{sm}}, \tag{68}$$

where

- $n$ is the number of targets in the scenario,
- $P_{ms}$ is the probability that a target will stop given that it is moving,
- $P_{sm}$ is the probability that a target will start moving given that it is stopped.

We select the desired average number of targets stopped by setting

$$P_{sm} = aP_{ms}, \tag{69}$$

for appropriate value of the scalar $a$. The transition probability $P_{ms}$ satisfies

$$P_{ms} = 1 - P_{mm}, \tag{70}$$

where $P_{mm}$ is the probability that the target will be moving given that it is currently moving. This probability is computed using the expected time a target will be moving given that it is moving, i.e.,

$$ExpectedTimeMoving = \{\text{time target will spend moving} \mid \text{target is currently moving}\} = \frac{P_{mm}}{1 - P_{mm}}\delta.$$

where $\delta$ is the time interval between two successive transitions of Markov chain modeling target motion mode. Using this relation, we can see that the transition probability $P_{mm}$ is given by

$$P_{mm} = \frac{ExpectedTimeMoving}{ExpectedTimeMoving + \delta}. \tag{71}$$

To summarize, the tracker parameters characterizing the target kinematic and motion models are as follows:

1. *Root mean square target velocity*. This velocity is used to compute the process noise covariance for the filter corresponding to the "move" mode over the time increment $\Delta Time$, as follows:

$$ProcessNoiseVariance = (RootMeanSquareVelocity \cdot \Delta Time)^2. \tag{72}$$

   The process noise variance for the filter corresponding to the "stop" mode is zero (which follows from the preceding formula with zero root mean square velocity).

2. *Maximum variance*. This is the maximum error variance allowed before a target track is modeled as being dropped.

3. *Expected time moving*. This is the expected time a target will be moving given that the target is currently moving. It is used for estimating the probability of the target of transitioning from the "move" to the "move" state as given in Equation (71).

**Sensor Model Parameters**

We have modeled two sensor modes, MTI and FTI. The MTI sensor mode can detect moving targets only, while the FTI sensor mode can detect stationary targets only. Both sensor modes are characterized by the following parameters:

1. *Detection probability*. For each sensor mode, the detection probabilities are currently fixed to a constant for all targets; however, the test setting allows one to model the scenarios where these probabilities are target dependent. The FTI detection probability depends on clutter and the number of successive looks. We use values (see **Error! Reference source not found.**) corresponding to moderate clutter and coarse image processing (a single look).

2. *Measurement error variance*. For each sensor mode, the sensor measurement errors are assumed to be Gaussian random variables with zero mean and unit standard deviation.

3. *Dwell-time*. This is the time a sensor takes to collect data in a particular mode.

**Dynamic Programming Parameters**

The parameters used in the dynamic programming formulation of the move-stop tracking problem are the following:

1. *Target priority values*.
2. *Discount factors*. There are two discount factors, one used in the reward-based formulation, and the other used in the cost-based formulation.
3. *Desired target error value*. For each target *I*, a desired position error variance $G_i$ is specified, which is used to define the goal region for the kinematic state of target *i*. The goal region is $\{S_i \mid S_i \leq G_i\}$.

### 4.4.2 Simulation Results

Our simulations are generated using the tracker and sensor parameter values given in Table 1 and Table 2, respectively. The parameter values used in the dynamic programming formulation of the SRM problem are listed in Table 3.

**Table 1: Tracker model parameter values used in our simulations**

| | |
|---|---|
| **Root mean square target velocity** | 10 meters per second |
| **Maximum variance** | 2500 square meters |
| **Expected time moving** | 1 minute |

**Table 2: Sensor model parameter values used in our simulations**

| | **MTI mode** | **FTI mode** |
|---|---|---|
| **Detection probability** | 0.9 | 0.9 |
| **Standard deviation of measurement error** | 1 meter | 1 meter |
| **Dwell-time** | 0.1 second | 10 seconds |

**Table 3: Dynamic programming parameter values used in our simulations**

| | |
|---|---|
| **Target priority value** | 1 (for all targets) |
| **Target goal state** | 5 meters (for all targets) |
| **Precision maximizing SRM discount factor, $\alpha$** | 0.65 |
| **Error minimizing SRM discount factor, $\gamma$** | 1 |

We next present the simulation results obtained for the problem of tracking 50 targets with a single sensor. We have four truth scenarios for the target motion that differ in the average number of targets stopped in the steady state. In particular, the average number of targets stopped is varying across the values 10, 20, 30, and 40. For each of these values, the transition

probabilities $P_{sm}$ and $P_{ss}$ for Markov chain modeling target motion are computed according to Equations (68) and (69) with $n$=50. Table 4 shows the relations between $P_{ms}$ and $P_{sm}$ for the four truth-scenarios.

**Table 4: The relation for the transition probabilities $P_{sm}$ and $P_{ms}$ as the average number of targets stopped (in the steady state) takes values 10, 20, 30, and 40**

| **Average number of targets stopped** | **$P_{sm}$ relation with $P_{ms}$ (*c.f.* Eq. 68)** |
|---|---|
| 10 | $P_{sm} = 4P_{ms}$ |
| 20 | $P_{sm} = \frac{3}{2}P_{ms}$ |
| 30 | $P_{sm} = \frac{2}{3}P_{ms}$ |
| 40 | $P_{sm} = \frac{1}{4}P_{ms}$ |

In all scenarios, the targets have the same priority value (see Table 3), and the tracking time is 10 minutes.

#### 4.4.2.1 Target Truth and SRM Control Plots

In this section, we present the plots of the control decisions for the precision maximizing SRM, the error minimizing SRM, and the entropy-based SRM. The SRM decisions are given for the target motion scenarios where the average number of stopped targets is 10, 20, 30, and 40.

*Control Decisions for the Precision Maximizing SRM*

The following Figure 5, Figure 6, Figure 7, and Figure 8 show the true target motion and the control decisions of the precision maximizing SRM algorithm for the scenarios with 10, 20, 30, and 40 targets stopped on average, respectively. The control decisions correspond to a typical sample path generated by the algorithm.

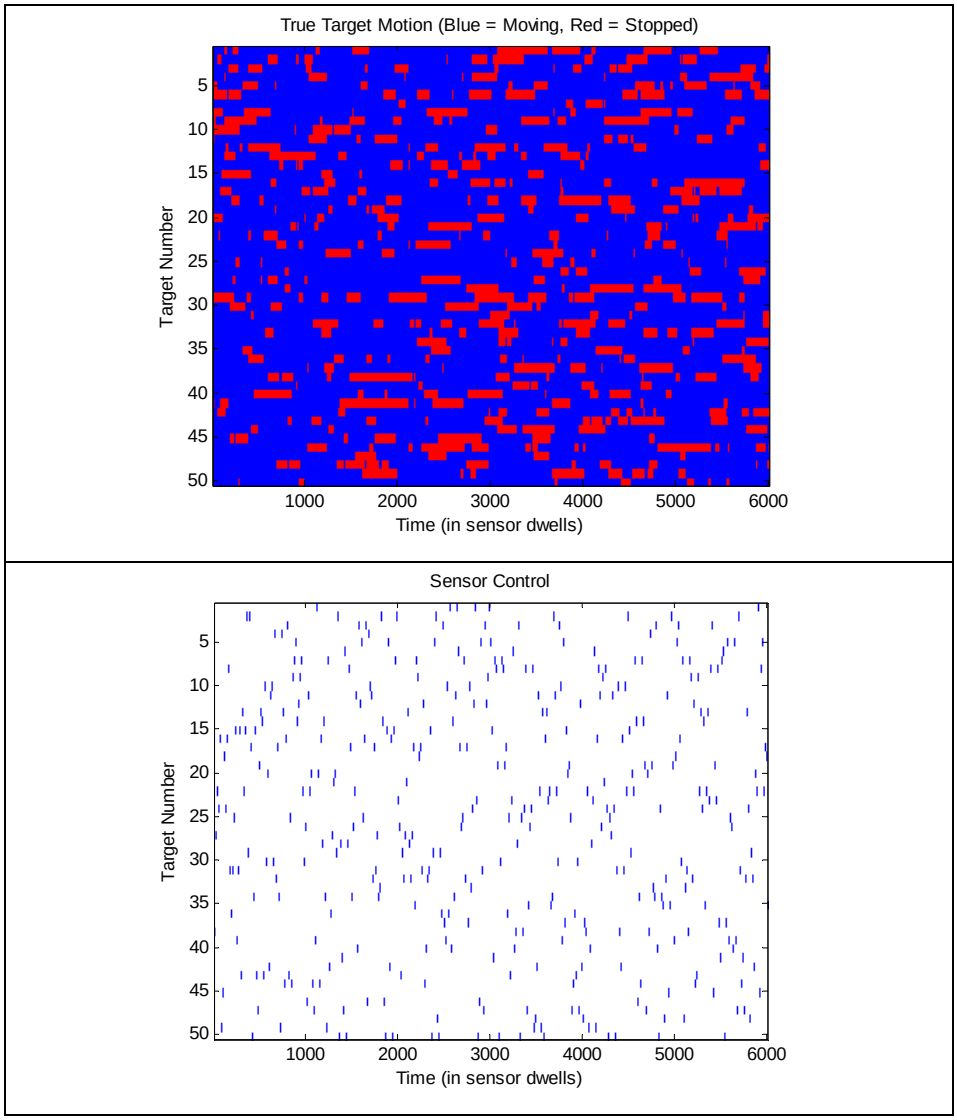


**Figure 5: The plots show results for the precision maximizing SRM when 10 targets are stopped on average. The top plot shows the true target motion. The blue color (dark shade) indicates that a target is moving and the red color (light shade) indicates that a target is stopped. For example, at the time corresponding to sensor dwell 1,000, target 1 is moving and target 2 is stopped. The bottom plot shows the control decisions corresponding to the precision maximizing SRM for the scenario with 10 targets stopped on average. The bars indicate which target is observed at which sensor dwell, and the bar color indicates the sensor mode used for the observation. The blue and the red colors correspond to the MTI and FTI sensor modes, respectively. For example, target 46 is observed at sensor dwell 1,000 in MTI mode. There are 6,000 sensor dwells scheduled during 10 minute tracking, and the MTI mode is used in each dwell.**

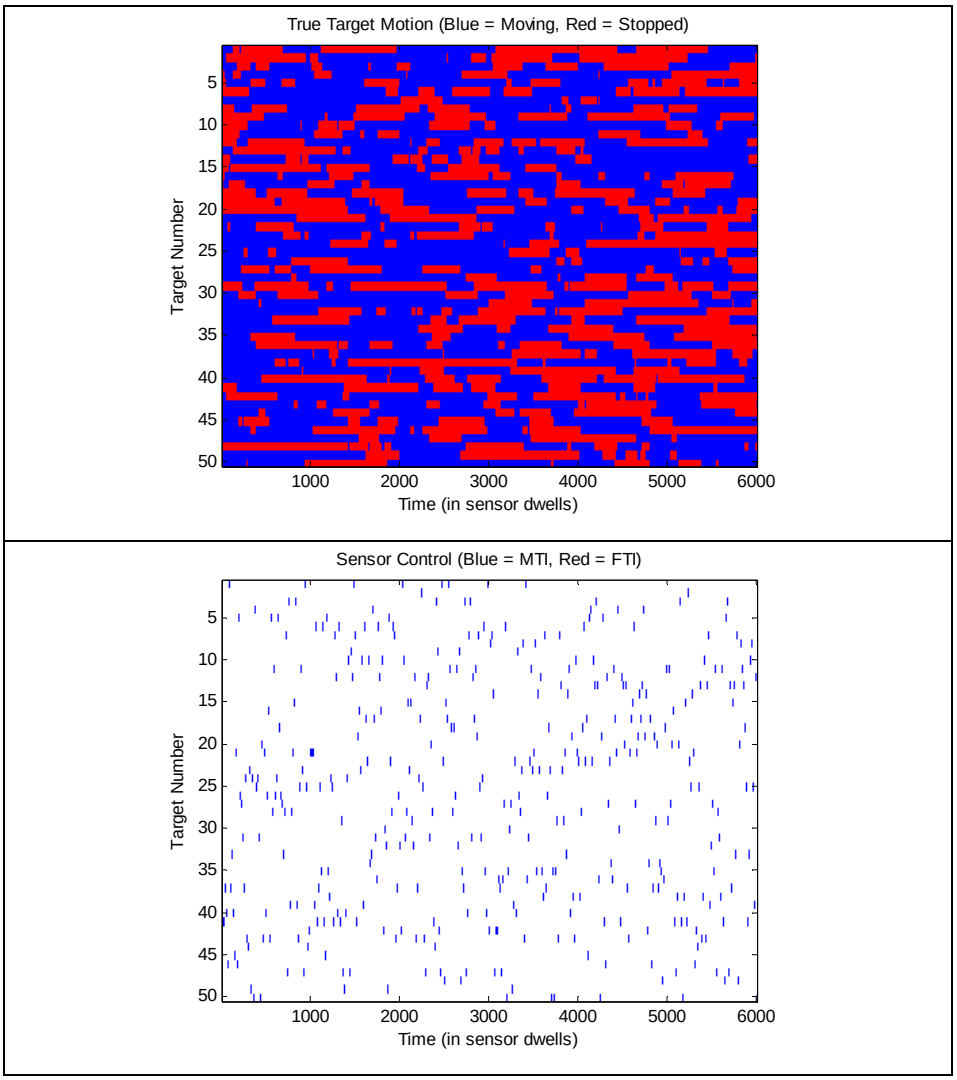


**Figure 6: The plots show results for the precision maximizing SRM when 20 targets are stopped on average. The top plot shows the true target motion. The blue color (dark shade) indicates that a target is moving and the red color (light shade) indicates that a target is stopped. For example, at the time corresponding to sensor dwell 1,000, target 1 is moving and target 2 is stopped. The bottom plot shows the control decisions corresponding to the precision maximizing SRM for the scenario with 20 targets stopped on average. The bars indicate which target is observed at which sensor dwell, and the bar color indicates the sensor mode used for the observation. The blue and the red colors correspond to the MTI and FTI sensor modes, respectively. For example, target 42 is observed at sensor dwell 1,000 in MTI mode. There are 6,000 sensor dwells scheduled during 10 minute tracking, and the MTI mode is used in each dwell.**

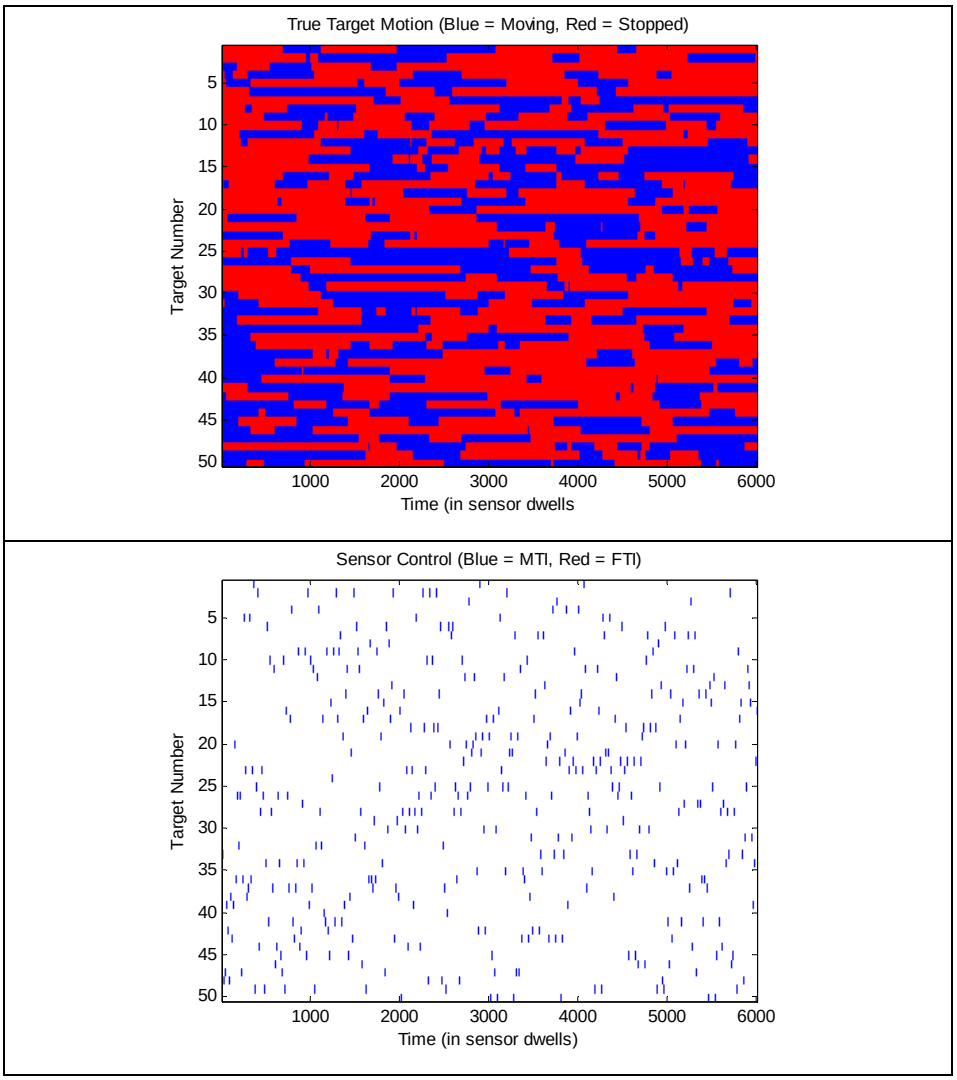


**Figure 7: The plots show results for the precision maximizing SRM when 30 targets are stopped on average. The top plot shows the true target motion. The blue color (dark shade) indicates that a target is moving and the red color (light shade) indicates that a target is stopped. For example, at the time corresponding to sensor dwell 1,000, target 1 is moving and target 2 is stopped. The bottom plot shows the the control decisions corresponding to the precision maximizing SRM for the scenario with 30 targets stopped on average. The bars indicate which target is observed at which sensor dwell, and the bar color indicates the sensor mode used for the observation. The blue and the red colors correspond to the MTI and FTI sensor modes, respectively. For example, target 37 is observed at sensor dwell 1,000 in MTI mode. There are 6,000 sensor dwells scheduled during 10 minute tracking, and the MTI mode is used in each dwell.**

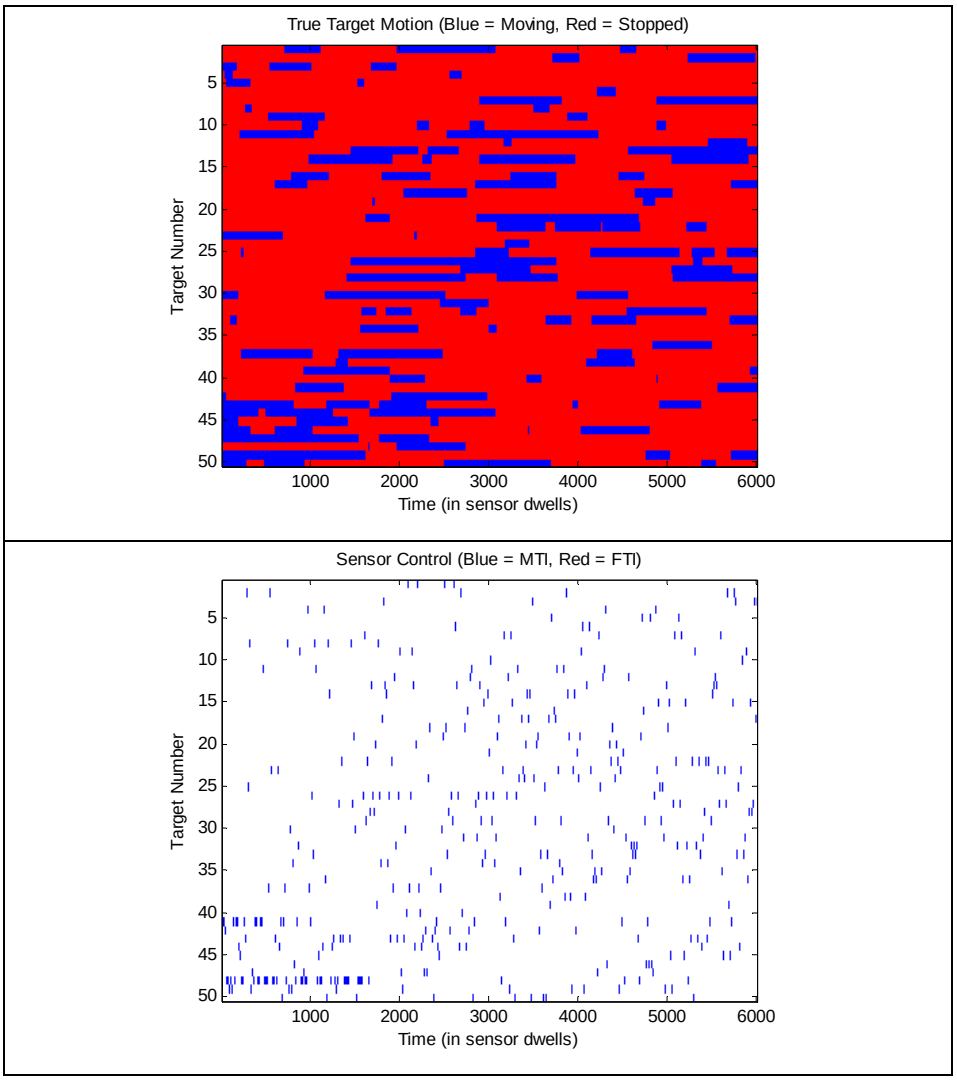


**Figure 8: The plots show results for the precision maximizing SRM when 40 targets are stopped on average. The top plot shows the true target motion. The blue color (dark shade) indicates that a target is moving and the red color (light shade) indicates that a target is stopped. For example, at the time corresponding to sensor dwell 1,000, target 1 is moving and target 2 is stopped. The bottom plot shows the control decisions corresponding to the precision maximizing SRM for the scenario with 40 targets stopped on average. The bars indicate which target is observed at which sensor dwell, and the bar color indicates the sensor mode used for the observation. The blue and the red colors correspond to the MTI and FTI sensor modes, respectively. For example, target 41 is observed at sensor dwell 1,000 in MTI mode. There are 6,000 sensor dwells scheduled during 10 minute tracking, and the MTI mode is used in each dwell.**

As indicated by the preceding plots, the precision maximizing SRM uses MTI mode exclusively. This is to be expected, since long FTI dwells are not beneficial for this SRM. More specifically, this SRM is maximizing the overall time the target errors are below the desired error goals. During the long FTI dwell, the errors of moving targets increase well above the desired error. Therefore, the time the target error goals are attained during a single FTI mode is shorter than the time the error goals are attained during a sequence of 100 MTI dwells. Being farsighted, the precision maximizing SRM captures the trade-off between the benefits resulting, respectively, from one long FTI dwell and a sequence of 100 short MTI dwells.

Note that the algorithm uses the MTI mode on stopped targets to check whether the target is still stopped or it has started moving.

*Control Decisions for the Error Minimizing SRM*

The following Figure 9, Figure 10, Figure 11, and Figure 13 show the true target motion and the control decisions of the error minimizing SRM algorithm for the scenarios with 10, 20, 30, and 40 targets stopped on average, respectively. The control decisions correspond to a typical sample path generated by this SRM algorithm.

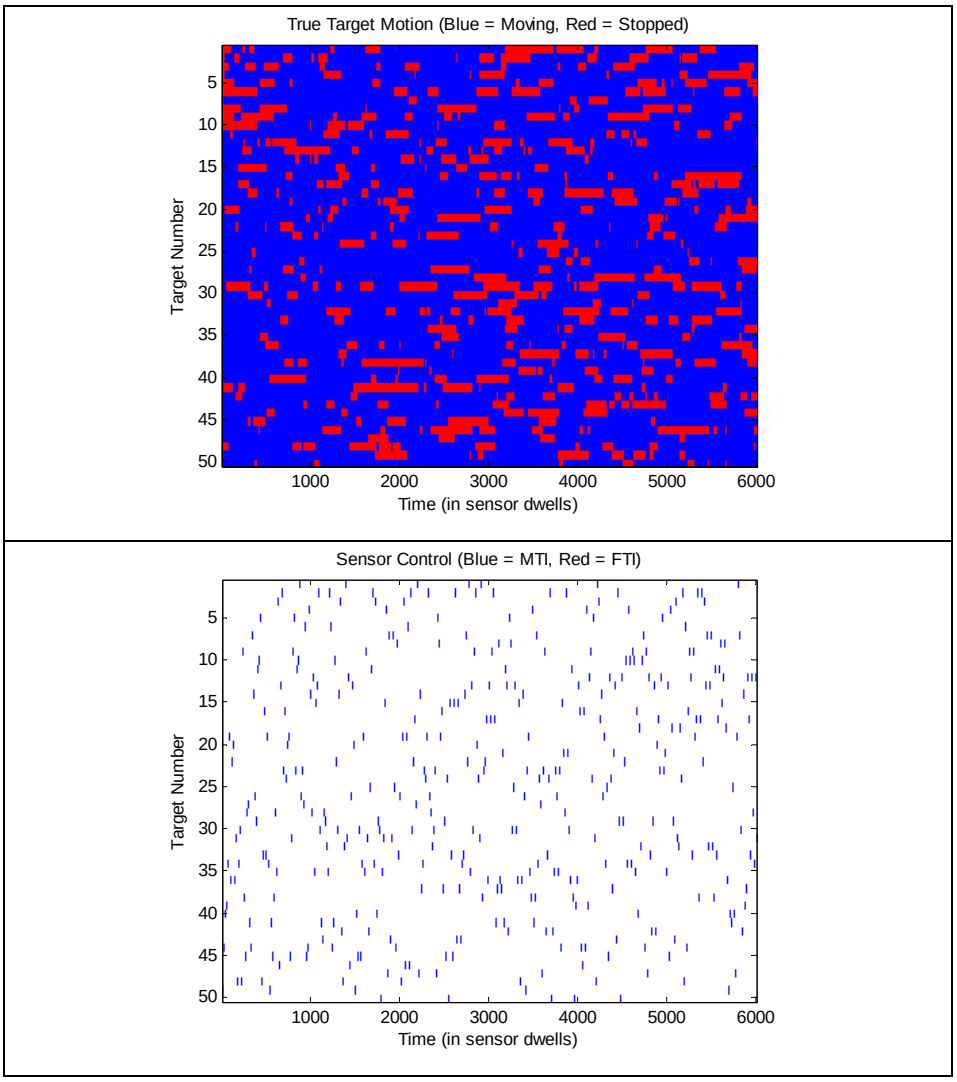


**Figure 9: The plots show results for the error minimizing SRM when 10 targets are stopped on average. The top plot shows the true target motion. The blue color (dark shade) indicates that a target is moving and the red color (light shade) indicates that a target is stopped. For example, at the time corresponding to sensor dwell 1,000, target 1 is moving and target 2 is stopped. The bottom plot shows the control decisions corresponding to the error minimizing SRM for the scenario with 10 targets stopped on average. The bars indicate which target is observed at which sensor dwell, and the bar color indicates the sensor mode used for the observation. The blue and the red colors correspond to the MTI and FTI sensor modes, respectively. For example, target 14 is observed at sensor dwell 1,000 in MTI mode. There are 6,000 sensor dwells scheduled during 10 minute tracking, and the MTI mode is used in each dwell.**

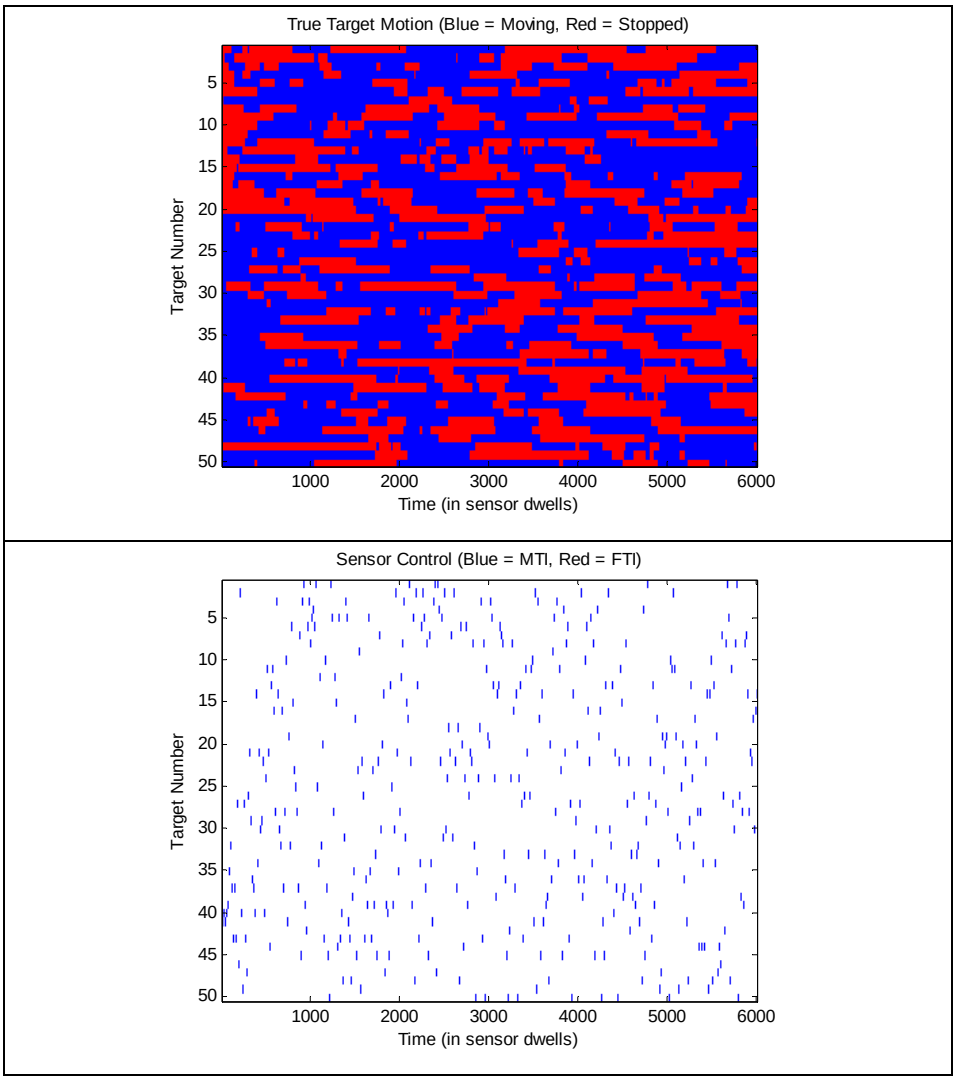


**Figure 10: The plots show results for the error minimizing SRM when 20 targets are stopped on average. The top plot shows the true target motion. The blue color (dark shade) indicates that a target is moving and the red color (light shade) indicates that a target is stopped. For example, at the time corresponding to sensor dwell 1,000, target 1 is moving and target 2 is stopped. The bottom plot shows the control decisions corresponding to the error minimizing SRM for the scenario with 20 targets stopped on average. The bars indicate which target is observed at which sensor dwell, and the bar color indicates the sensor mode used for the observation. The blue and the red colors correspond to the MTI and FTI sensor modes, respectively. For example, target 7 is observed at sensor dwell 1,000 in MTI mode. There are 6,000 sensor dwells scheduled during 10 minute tracking, and the MTI mode is used in each dwell.**

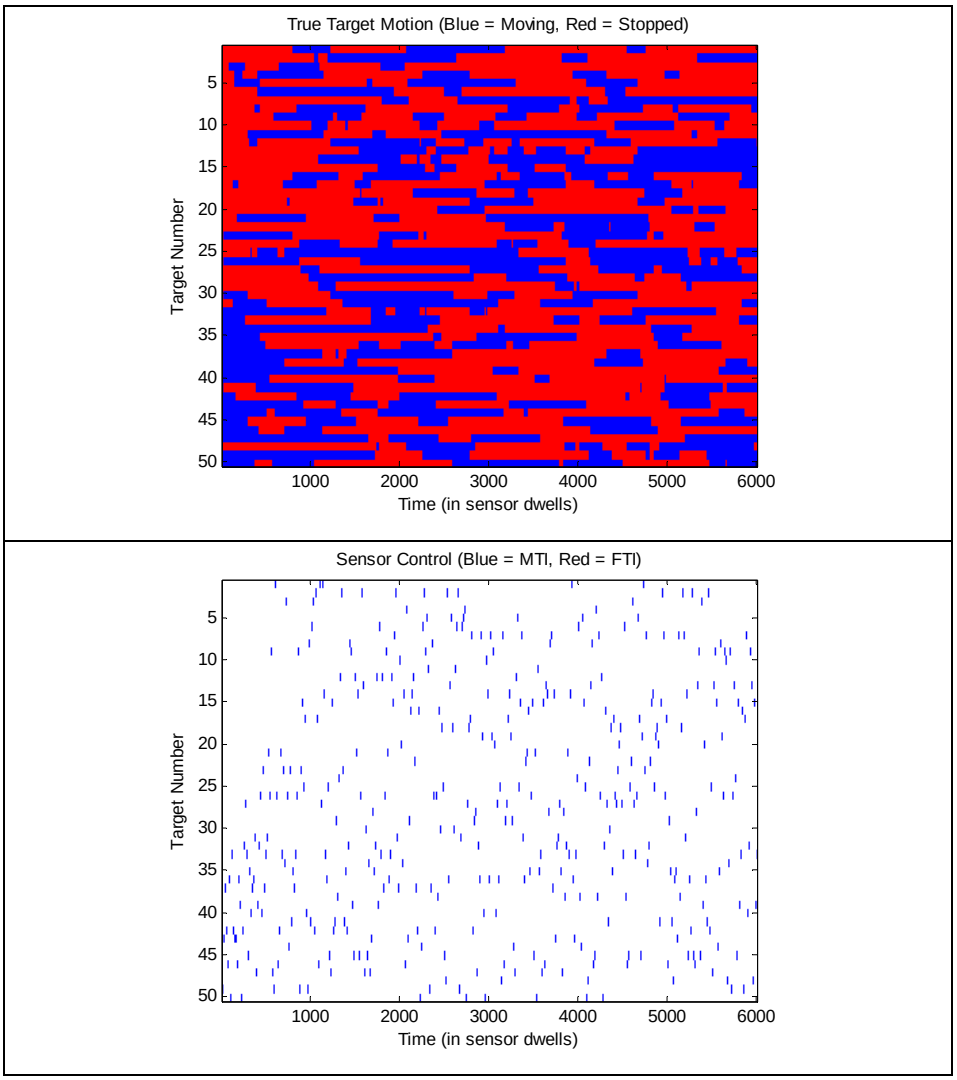


**Figure 11: The plots show results for the error minimizing SRM when 30 targets are stopped on average. The top plot shows the true target motion. The blue color (dark shade) indicates that a target is moving and the red color (light shade) indicates that a target is stopped. For example, at the time corresponding to sensor dwell 1,000, target 1 is moving and target 2 is stopped. The bottom plot shows the control decisions corresponding to the error minimizing SRM for the scenario with 30 targets stopped on average. The bars indicate which target is observed at which sensor dwell, and the bar color indicates the sensor mode used for the observation. The blue and the red colors correspond to the MTI and FTI sensor modes, respectively. For example, target 41 is observed at sensor dwell 1,000 in MTI mode. There are 6,000 sensor dwells scheduled during 10 minute tracking, and the MTI mode is used in each dwell.**

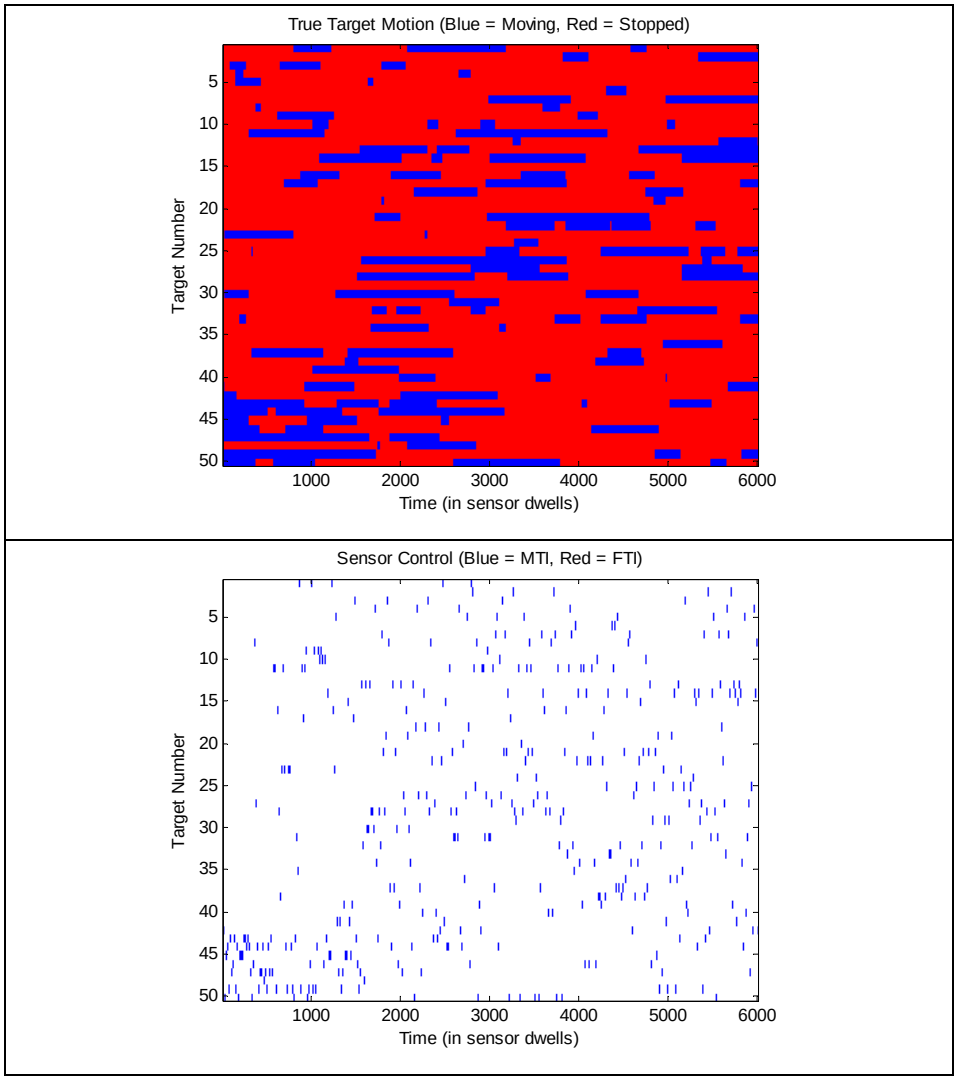


**Figure 12: The plots show results for the error minimizing SRM when 40 targets are stopped on average. The top plot shows the true target motion. The blue color (dark shade) indicates that a target is moving and the red color (light shade) indicates that a target is stopped. For example, at the time corresponding to sensor dwell 1,000, target 1 is moving and target 2 is stopped. The bottom plot shows the control decisions corresponding to the error minimizing SRM for the scenario with 40 targets stopped on average.The bars indicate which target is observed at which sensor dwell, and the bar color indicates the sensor mode used for the observation. The blue and the red colors correspond to the MTI and FTI sensor modes, respectively. For example, target 49 is observed at sensor dwell 1,000 in MTI mode. There are 6,000 sensor dwells scheduled during 10 minute tracking, and the MTI mode is used in each dwell.**

Similar to the precision maximizing SRM, the error minimizing SRM uses MTI mode exclusively as indicated in the preceding plots. Again, this is to be expected, since this SRM is minimizing the discounted sum of the target errors above the level of the desired error goal accumulated over the tracking time. The total target error accumulated during a long FTI dwell is larger than the target error accumulated during a sequence of 100 MTI dwells. Therefore, the FTI mode is less beneficial for the cost-rollout algorithm than the MTI mode. Being farsighted, the error minimizing SRM can anticipate the benefits resulting from a long FTI dwell and a sequence of 100 short MTI dwells, and can select a control that is more beneficial in a long run.

Note that, similar to the precision maximizing algorithm, the error minimizing uses the MTI mode for stopped targets to check whether the target is still stopped or it has started moving.

*Control Decisions for the Entropy-Based SRM*

The following Figure 13, Figure 14, Figure 15, and Figure 16 show the true target motion and the control decisions of the myopic entropy-based SRM for the scenarios with 10, 20, 30, and 40 targets stopped on average, respectively. The control decisions correspond to a typical sample path generated by this algorithm.

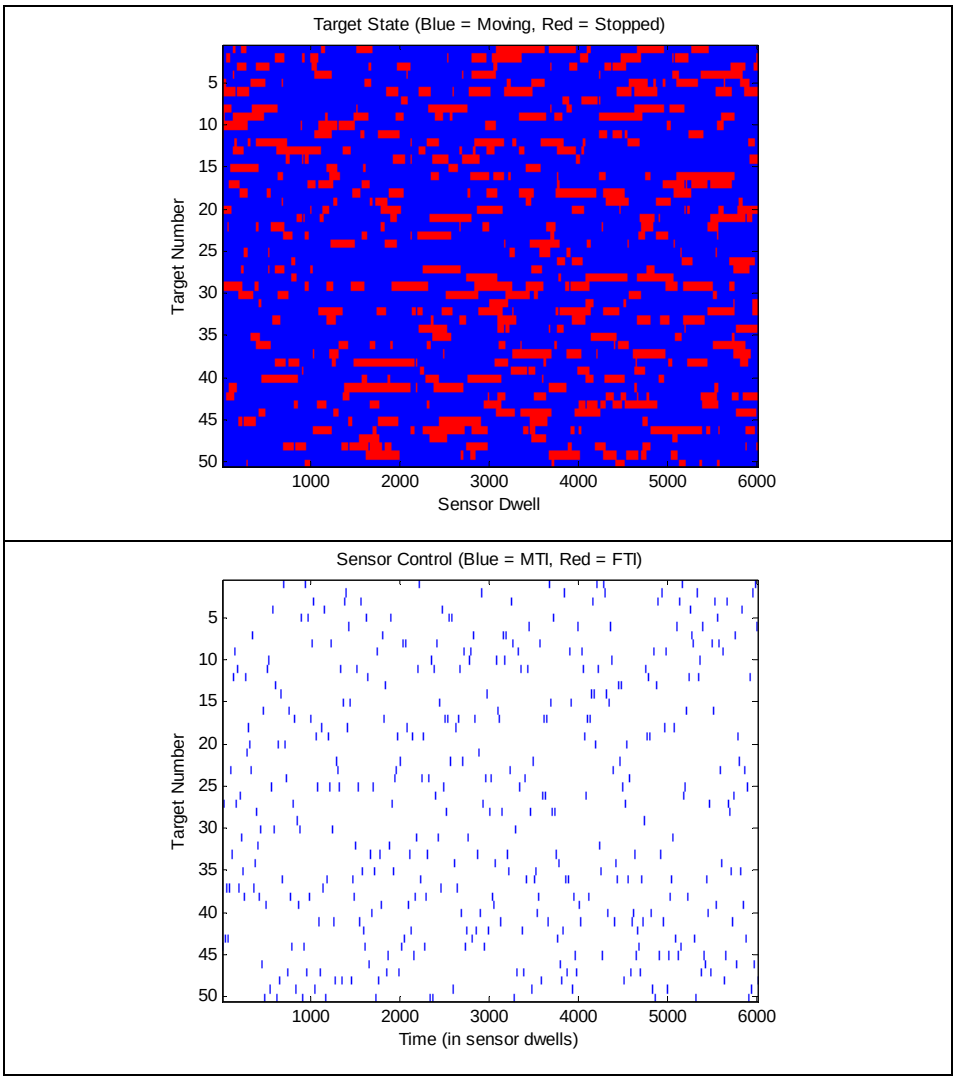


**Figure 13: The plots show results for the myopic entropy-based SRM when 10 targets are stopped on average. The top plot shows the true target motion. The blue color (dark shade) indicates that a target is moving and the red color (light shade) indicates that a target is stopped. For example, at the time corresponding to sensor dwell 1,000, target 1 is moving and target 2 is stopped. The bottom plot shows the control decisions corresponding to the entropy-based SRM for the scenario with 10 targets stopped on average. The bars indicate which target is observed at which sensor dwell, and the bar color indicates the sensor mode used for the observation. The blue and the red colors correspond to the MTI and FTI sensor modes, respectively. For example, target 37 is observed at sensor dwell 1,000 in MTI mode. During 10 minute tracking, 6,000 sensor dwells are scheduled and the MTI mode is used in each dwell.**

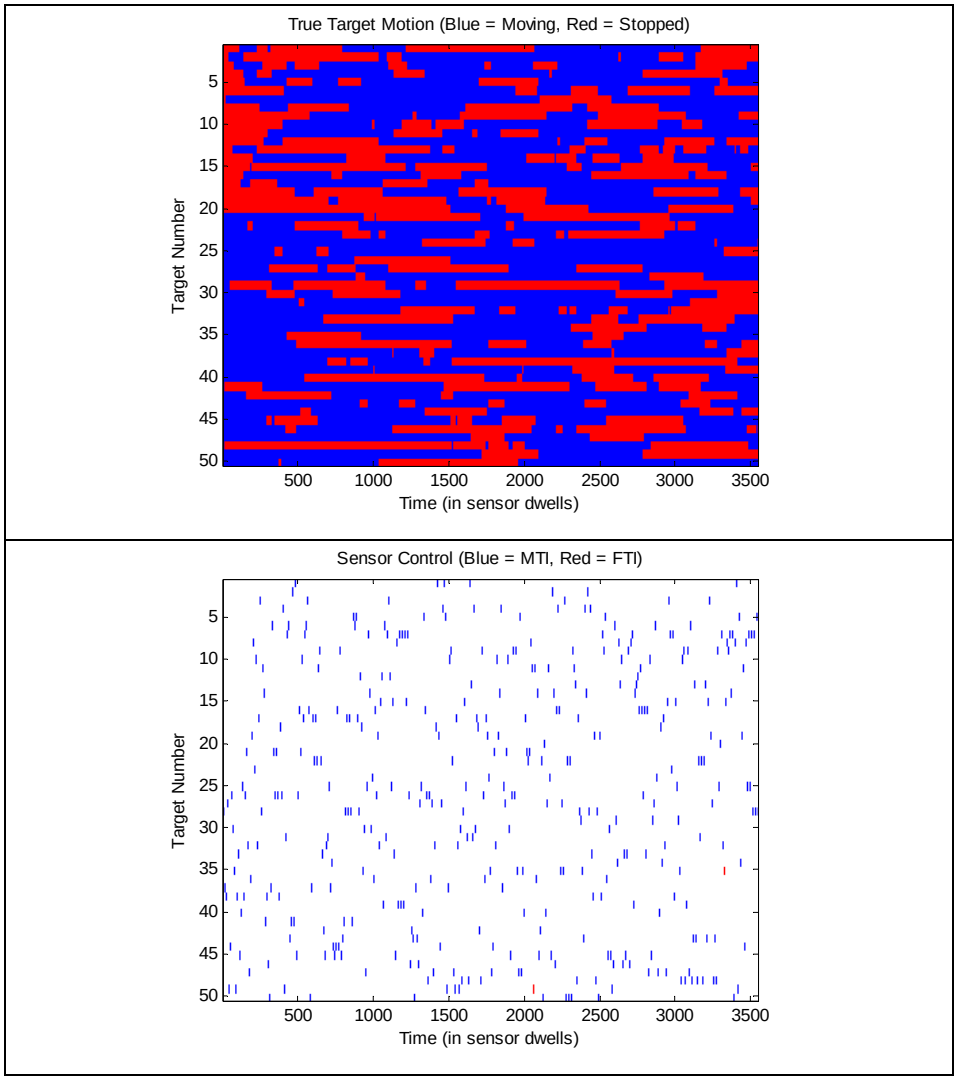


**Figure 14: The plots show results for the myopic entropy-based SRM when 20 targets are stopped on average. The top plot shows the true target motion. The blue color (dark shade) indicates that a target is moving and the red color (light shade) indicates that a target is stopped. For example, at the time corresponding to sensor dwell 500, target 1 is stopped and target 4 is moving. The bottom plot shows the control decisions corresponding to the entropy-based SRM for the scenario with 20 targets stopped on average.The bars indicate which target is observed at which sensor dwell, and the bar color indicates the sensor mode used for the observation. The blue and the red colors correspond to the MTI and FTI sensor modes, respectively. For example, target 45 is observed at sensor dwell 500 in MTI mode and target 49 is observed at sensor dwell 2,052 in FTI mode. During 10 minute tracking, 3,546 sensor dwells are scheduled and the long FTI mode is used in 25 dwells.**

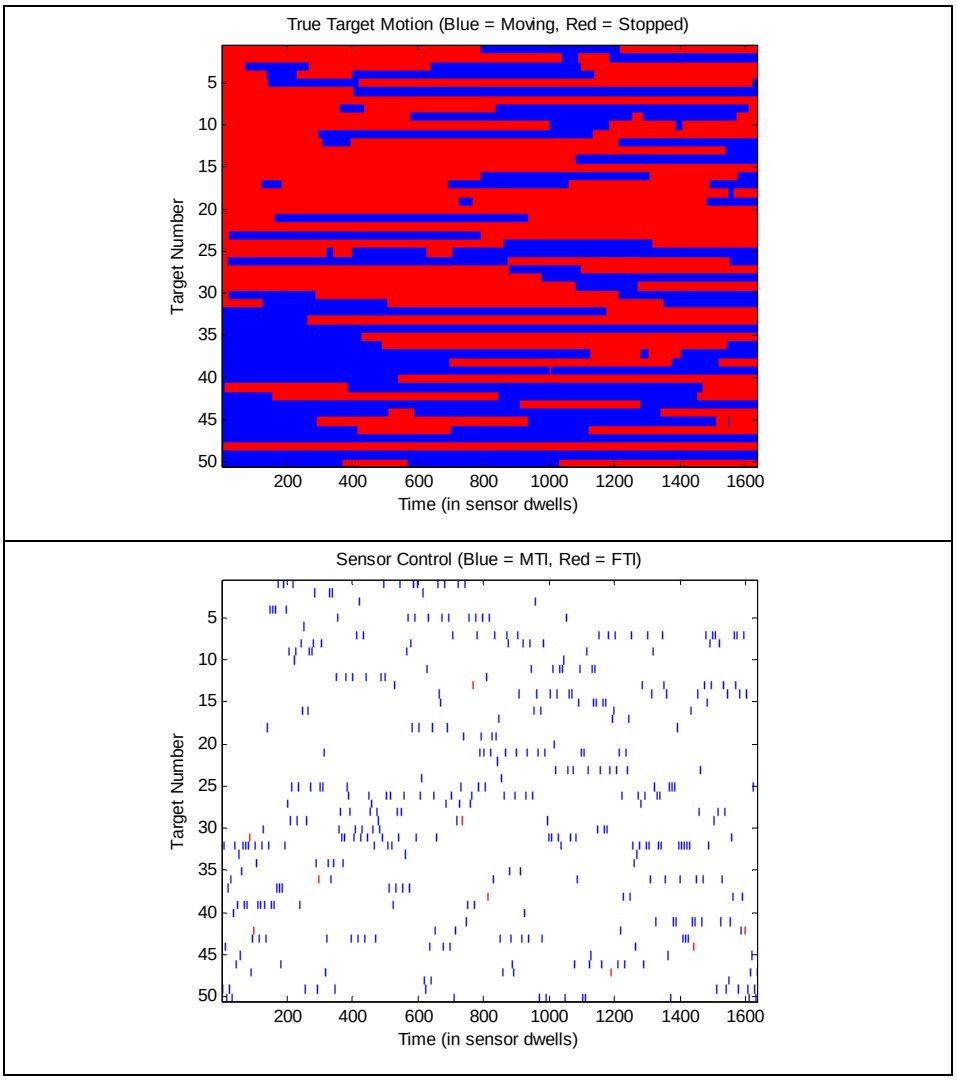


**Figure 15: The plots show results for the myopic entropy-based SRM when 30 targets are stopped on average. The top plot shows the true target motion The blue color (dark shade) indicates that a target is moving and the red color (light shade) indicates that a target is stopped. For example, at the time corresponding to sensor dwell 200, target 1 is stopped and target 3 is moving. The bottom plot shows the control decisions corresponding to the entropy-based SRM for the scenario with 30 targets stopped on average. The bars indicate which target is observed at which sensor dwell, and the bar color indicates the sensor mode used for the observation. The blue and the red colors correspond to the MTI and FTI sensor modes, respectively. For example, target 29 is observed at sensor dwell 200 in MTI mode and target 40 is observed at sensor dwell 110 in FTI mode. During 10 minute tracking, 1,633 sensor dwells are scheduled and the long FTI mode is used in 45 dwells.**

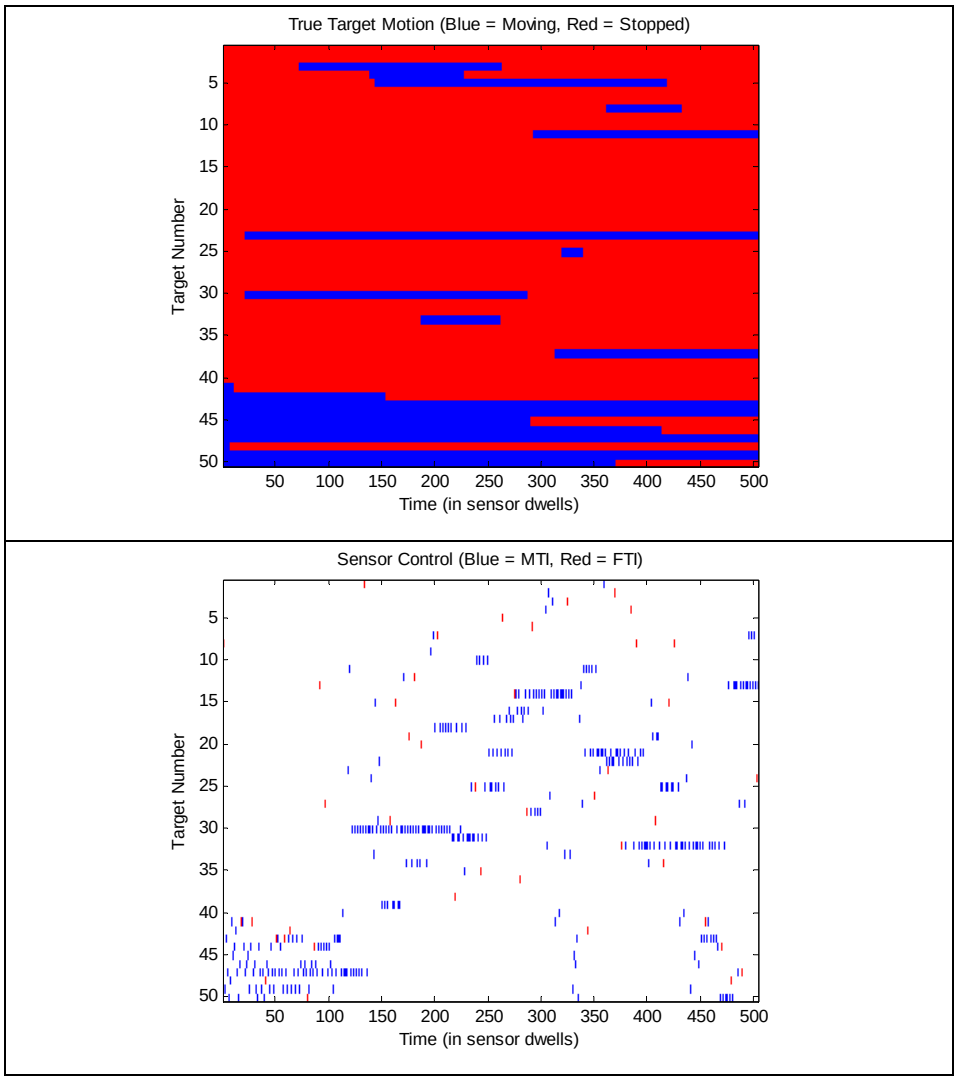


**Figure 16: The plots show results for the myopic entropy-based SRM when 40 targets are stopped on average. The top plot shows the true target motion. The blue color (dark shade) indicates that a target is moving and the red color (light shade) indicates that a target is stopped. For example, at the time corresponding to sensor dwell 50, target 1 is stopped and target 30 is moving. The bottom plot shows the control decisions corresponding to the entropy-based SRM for the scenario with 40 targets stopped on average. The bars indicate which target is observed at which sensor dwell, and the bar color indicates the sensor mode used for the observation. The blue and the red colors correspond to the MTI and FTI sensor modes, respectively. For example, target 44 is observed at sensor dwell 50 in MTI mode and target 48 is observed at sensor dwell 42 in FTI mode. During 10 minute tracking, 504 sensor dwells are scheduled and the long FTI mode is used in 56 dwells.**

As seen from the preceding plots Figure 13 through Figure 16, the entropy-based SRM algorithm schedules the longer FTI mode more frequently to observe the stopped targets, as the number of stopped targets increases. In particular:

1. When 10 targets are stopped on average, the FTI mode is not used in any dwell.
2. When 20 targets are stopped on average, the FTI mode is used in 25 dwells out of the total of 3546 dwells scheduled during the 10 minute tracking.
3. When 30 targets are stopped on average, the FTI mode is used in 45 dwells out of the total of 1633 dwells scheduled during the 10 minute tracking.
4. When 40 targets are stopped on average, the FTI mode is used in 56 dwells out of the total of 504 dwells scheduled during the 10 minute tracking.

The entropy-based SRM algorithm, being myopic, does not anticipate long term benefits resulting from current control decisions. In particular, this algorithm selects control decisions based on the benefits of a single dwell plan. For such a short planning horizon, the FTI mode may seem more beneficial than the MTI mode, which results in scheduling dwells with the FTI mode.

### 4.4.3 Time Averaged Mean-Square Error and Fraction of Time the Error Goals are Met

Here, we present our simulation results for the farsighted precision maximizing SRM and the error minimizing SRM algorithms, and the myopic entropy-based SRM algorithm obtained for the four target-motion scenarios. We have 4 Monte Carlo simulations for each target-motion scenario and each SRM algorithm. Note that the number of Monte Carlo simulations needed to achieve reasonably small 95 percent confidence intervals is small because there is not much variability in the simulation results. In particular, all the variability in the simulation results (for a fixed scenario and a selected SRM algorithm) comes from the uncertainty in the sensor detections and the sensor detection probability is 0.9 for both modes (see Table 2).

We present our simulation results in terms of two measures of performance:

1. Time averaged mean-square error.
2. Average fraction of time the target error goals are met.

The time averaged mean-square error is computed by averaging the sum of target errors in time over the number of targets, over the tracking time, and over the number of runs. Similarly, the average fraction of time the target error goals are met is computed by averaging the sum of the fractions of time the target error goal is met over the number of targets and over the number of runs. The fraction of time the target error goal is met is the total time (during the tracking) the target error is below the desired error goal divided by the total tracking time. The simulation results for the time averaged mean-square error are presented in Figure 17, and the results for the average fraction of time the error goals are met are shown in Figure 18. The bars in the figure mark the intervals for the variability in the data with 95 percent confidence.

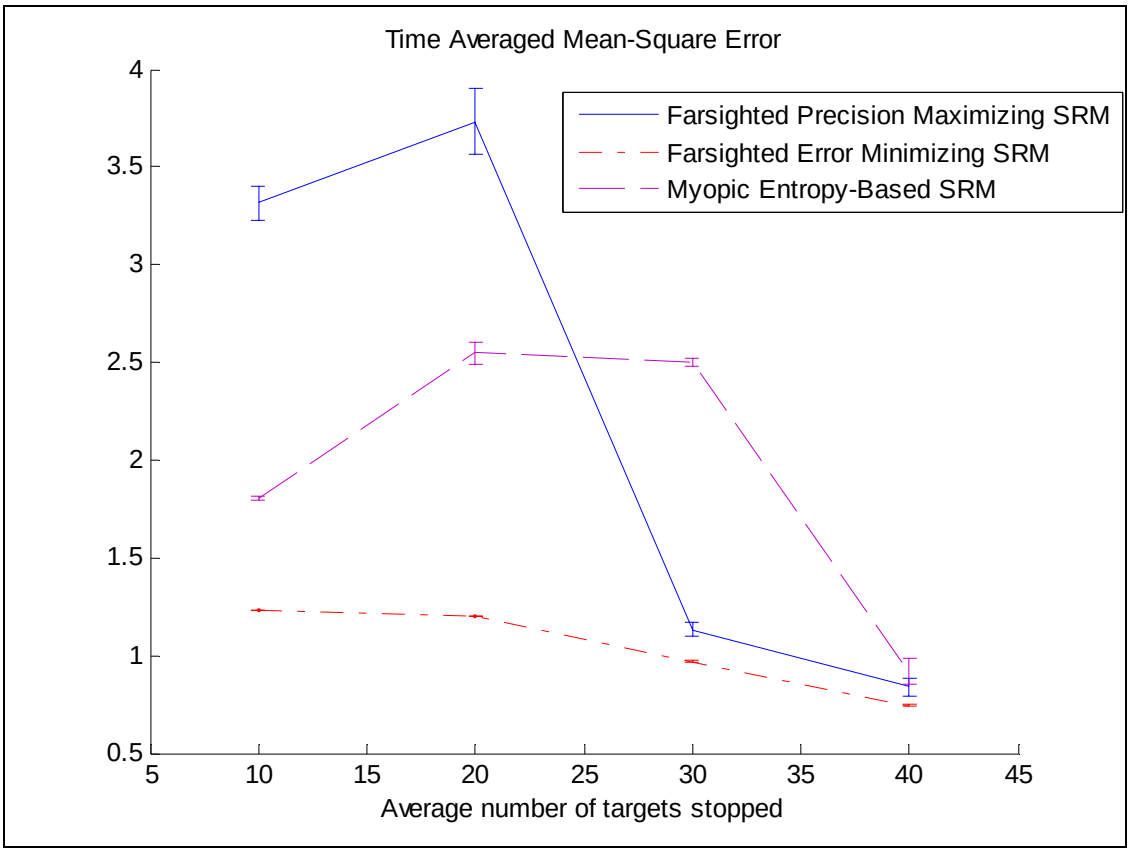


**Figure 17: The results for the time averaged mean-square error obtained for the four target-motion scenarios and for the three SRM algorithms.**

The four target-motion scenarios correspond to the average number of targets stopped taking values 10, 20, 30, and 40, respectively. The three SRM algorithms are the farsighted precision maximizing and error minimizing SRM algorithms, and the myopic entropy-based SRM algorithm.

The simulation results in Figure 17 indicate that the farsighted error minimizing SRM algorithm maintains better quality target tracks than the farsighted precision maximizing and the myopic algorithms. In particular, the difference in the time averaged mean-square error for the precision maximizing SRM and for the error minimizing SRM is largest when most of the targets are moving (see the results in Figure 17 for 10 and 20 targets stopped on average). When most of the targets are moving, the entropy-based algorithm also has smaller error than the precision maximizing algorithm. However, as more targets are stopped (see the results for 30 targets stopped on average), the entropy-based algorithm accumulates more error than the precision maximizing algorithm. When most of the targets are stopped, the tracking is easier because there

are fewer constraints on the sensor resources. In this case, all three algorithms perform similarly (see the results for 40 targets stopped on average).

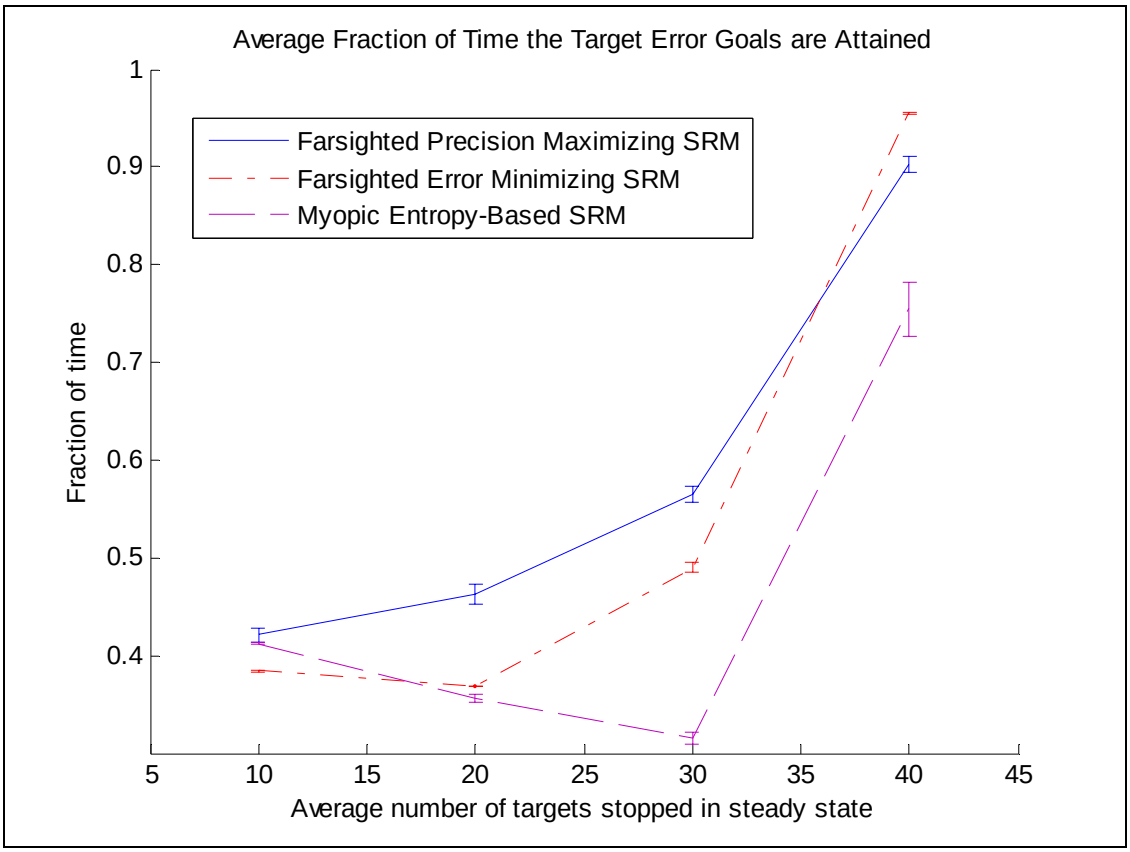


**Figure 18: The results for the average fraction of time the target error goals are met obtained for the four target-motion scenarios and for the three SRM algorithms.**

The four target-motion scenarios correspond to the average number of targets stopped taking values 10, 20, 30, and 40, respectively. The three SRM algorithms are the farsighted precision maximizing and error minimizing SRM algorithms, and the myopic entropy-based SRM algorithm.

The simulation results in Figure 18 indicate that the farsighted precision maximizing SRM algorithm has the best performance and the myopic entropy-based has the worst performance for the fraction of time the target error goals are met. The difference in the fraction of time the error goals are met between the precision maximizing SRM and the error minimizing SRM is the largest when most of the targets are moving (see the results in Figure 17 for 20 targets stopped on average). The difference in the fraction of time the error goals are met between the precision

maximizing and the entropy-based algorithm is the largest when most of the targets are stopped (see the results in Figure 18 for 30 targets stopped on average).

Considering the results for the precision maximizing algorithm in both Figure 17 and Figure 18, and the fact that this algorithm has scheduled the MTI mode for each dwell, it seems that the precision maximizing algorithm keeps the target error below the desired error goal for as many targets as possible, resulting in a very small error for some targets and very large error on the other targets. Consequently, the average target error is large, but the overall time the error goals are met is also large.

Considering the results for the error minimizing algorithm in both Figure 17 and Figure 18, we see that this algorithm maintains overall small errors on all targets, but does not necessarily maintain the errors smaller than their corresponding error goals. Consequently, this algorithm has small average target error, but the overall time the error goals are met is not always the best.

The performance of the myopic entropy-based algorithm is not satisfactory in either of the two measures. This algorithm schedules the longer FTI mode more frequently to observe the stopped targets as the number of stopped targets increases (see Figure 16). This results in substantially less time spent observing moving targets, and the corresponding track errors far exceed the goals. Consequently, the overall target error is large and the time target error goals are met is short.

## 4.5 CONCLUSIONS

We have developed novel, computable, farsighted SRM algorithms for move/stop tracking with a multi-mode sensor. This particular sensor management problem is challenging because of the complex target dynamics and the variable duration of sensor actions. We have evaluated the farsighted algorithms against a myopic, entropy-based SRM algorithm. Our simulation results indicate that the farsighted algorithms have promising behavior. For example, the farsighted precision maximizing SRM results in a policy that adaptively adjusts the frequency with which moving and stopped targets are observed in a manner that results in better tracks than the myopic entropy-based sensor manager. We attribute this to the capability of the farsighted algorithm to adapt the target revisit rates appropriately.

We believe that our simulation results are important indicators that the farsighted algorithms are better than myopic ones, especially, for SRM problems with complex dynamics (e.g., when targets are randomly starting and stopping and/or sensor actions have significantly different durations).

# 5 COMPUTABLE OPTIMAL STRATEGIES

In this section, we investigate computable optimal strategies for sensor resource management (SRM) problems. SRM problems can be formulated as Markov decision processes (MDPs) which in turn can be solved optimally, at least in principle, by numerical dynamic programming algorithms. Since the processing time and memory required to solve the dynamic program associated with the MDP in SRM is exponential in the number $n$ of targets being sensed, optimal numerical solutions of the general SRM problem are intractable for large $n$ of interest (e.g., for hundreds of targets). However, there are classes of problems such as multi-armed bandit problems which have optimal strategies in terms of maximizing a priority index rule computed independently for each target. These strategies are computationally tractable, and can be used as subroutines in computing approximate optimal strategies of more realistic problems. Sometimes priority index solutions can be obtained for problems which aren't multi-armed bandit problems. For example, it is shown in [7] that a priority index solution based on the conditional probabilities of each target being a threat is optimal for a finite horizon classification problem.

This report describes a sufficient condition to use for checking whether a given strategy, such as one given by a priority index rule, is optimal. The sufficient condition applies to finite horizon MDPs with terminal reward, and can be used to show the optimality of the search strategy in [7] and in some other examples that we will describe. The report is organized as follows:

- Section 5.1 formulates the sufficient condition for a general MDP in terms of strategy sets. This section defines strategy set, terminal optimality of a strategy set, deferrable decisions, and commutative transition probabilities. If a strategy set is terminally optimal, has deferrable decisions, and the MDP has commutative transition probabilities, then the strategy set is optimal. The section specializes the general result to symmetric MDP problems, which are given in some of the examples later in the section.
- Section 5.2 describes a subclass of MDP that corresponds to many SM problems, namely the class of MDP corresponding to one sensor observing $n$ non-interacting targets one at a time. This section specializes the definitions and results of Section 5.1 to this subclass of SM MDP problems.
- Section 5.3 applies the sufficient condition to show the optimality of a strategy for a class of MDP (which is a subclass of the SM MDP problems of Section 5.2). This class of MDP is characterized by $n$ non-interacting Markov chains which have an ordered state space. In particular, the chains can only transition at most one step in one direction and include birth-death processes as a special case.
- Section 5.4 applies the sufficient condition to show the optimal strategy for a class of symmetric binary classification problems.
- Section 5.5 applies the sufficient condition to show the optimal strategy for a class of search problems. This strategy was developed previously in [7].

## 5.1 SUFFICIENT CONDITION

We will denote a MDP with terminal reward by the tuple $(\mathsf{X}, \mathsf{U}, p_u, R, T)$ where $\mathsf{X}$ denotes the state space of the Markov chain, $\mathsf{U}$ denotes the set of possible decisions, $\{p_u : u \in \mathsf{U}\}$ is the collection of transition probabilities parameterized by the decision $u$, $R$ is the terminal reward function $R : \mathsf{X} \to \mathsf{R}$, and the integer $T$ is the terminal time. We will assume that $\mathsf{X}$ is discrete and $\mathsf{U}$ is finite.

If $X(t), 0 \le t \le T$ is the Markov process with decisions $U(t), 0 \le t \le T-1$, and terminal reward $R(X(T))$, the MDP problem is to select $U$ to maximize the expected value $\mathrm{E}\{R(X(T))\}$ of the terminal reward. We assume that the decision $U(t)$ depends only on $X(0),..., X(t)$ and that

$$\Pr\{X(t+1) = \xi \mid X(t) = x, U(t) = u\} = p_u(\xi \mid x). \tag{72}$$

The dynamic programming equations for the optimal reward function for the MDP $(\mathsf{X}, \mathsf{U}, p_u, R, T)$ are given as follows. The terminal condition is

$$V(x, T) = R(x). \tag{73}$$

The recursion is

$$V(x, t) = \max_u \{V_u(x, t)\} \tag{74}$$

for times $0 \le t \le T-1$, where we define

$$V_u(x, t) := \sum_\xi V(\xi, t+1)\, p_u(\xi \mid x). \tag{75}$$

**Definition 1.** Suppose that the MDP $(\mathsf{X}, \mathsf{U}, p_u, R, T)$ has the probability transition functions $p_u(\xi \mid x)$ for $x, \xi \in \mathsf{X}$, $u \in \mathsf{U}$, and terminal reward $R(x)$ for $x \in \mathsf{X}$. If $\Phi(x, t) \subset \mathsf{U}$ for each $x \in \mathsf{X}$ and $0 \le t \le T-1$, we say that $\Phi$ is a **strategy set** for the MDP.

**Definition 2.** If $\Phi$ is a strategy set for the MDP $(\mathsf{X}, \mathsf{U}, p_u, R, T)$, and if for each $x \in \mathsf{X}$ and each $u \in \Phi(x, T-1)$,

$$\sum_y R(y)\, p_u(y \mid x) = \max_v \sum_y R(y)\, p_v(y \mid x), \tag{76}$$

we say that the strategy set $\Phi$ is **terminally optimal** for the MDP.

**Definition 3.** If $\Phi$ is a strategy set for the MDP $(\mathsf{X}, \mathsf{U}, p_u, R, T)$, and if for each $t$ such that $0 \le t \le T-1$, each $x \in \mathsf{X}$, and for all $u, v, y$

$$u \in \Phi(x,t),\ V_v(x,t) > V_u(x,t),\ \text{and}\ \ p_v(y \mid x) > 0 \quad \text{imply that} \quad u \in \Phi(y, t+1), \tag{77}$$

then we say that **decisions are deferrable** in the strategy set $\Phi$.

**Remark 1.** Definition 3 gives conditions under which if $u$ is in the decision set at the current time but a different decision $v$ is made, then $u$ is still in the decision set at the next time. This condition allows using an interchange argument to prove the optimality of the decision set (Theorem 1). Unfortunately, Definition 3 is too hard to check in practice. However, it is implied by various stronger conditions that are easier to check. For example, if for each $t$ such that $0 \le t \le T-1$, and each $x \in \mathsf{X}$,

$$u \in \Phi(x,t),\ v \ne u,\ \text{and}\ \ p_v(y \mid x) > 0 \quad \text{imply that} \quad u \in \Phi(y, t+1), \tag{78}$$

then decisions are deferrable in the strategy set $\Phi$. This condition is stronger than the definition, since $V_v(x,t) > V_u(x,t)$ obviously implies that $v \ne u$. At the end of this section we prove another stronger condition for problems with symmetry.

**Definition 4.** We say that the probability transition functions $p_u(\xi \mid x)$ are **commutative** if for all $u, v \in \mathsf{U}$,

$$\sum_{\eta} p_u(\xi \mid \eta)\, p_v(\eta \mid x) = \sum_{\eta} p_v(\xi \mid \eta)\, p_u(\eta \mid x) \tag{79}$$

for all $x, \xi \in \mathsf{X}$.

**Theorem 1.** Suppose that $\Phi$ is a strategy set for an MDP $(\mathsf{X}, \mathsf{U}, p_u, R, T)$ with commutative transition probability functions $p_u$, such that $\Phi$ is terminally optimal and decisions in $\Phi$ are deferrable. Then the strategy set $\Phi$ is optimal in the sense that any decision $U(t) \in \Phi(X(t), t)$ for $0 \le t \le T-1$, is an optimal decision for $(\mathsf{X}, \mathsf{U}, p_u, R, T)$.

*Proof.* Define $\Phi^*(x,t)$ to be the set

$$\Phi^*(x,t) = \left\{ u \ :\ V_u(x,t) = \max_w V_w(x,t) \right\} \tag{80}$$

Thus, $\Phi^*(x,t)$ is the set of optimal strategies. We want to prove that

$$\Phi(x,t) \subset \Phi^*(x,t). \tag{81}$$

The terminal optimality condition is equivalent to

$$\Phi(x,T-1) \subset \Phi^*(x,T-1). \tag{82}$$

Thus, assume that $\Phi(x,t+1) \subset \Phi^*(x,t+1)$ is true for $\Phi$ and prove (81) from it. Suppose that $u \in \Phi(x,t)$ and $u \notin \Phi^*(x,t)$. Clearly $\Phi^*(x,t) \neq \varnothing$ and there is $v \in \Phi^*(x,t)$ such that $V_v(x,t) > V_u(x,t)$. The condition that decisions in $\Phi$ are deferrable implies that $u \in \Phi(X(t+1),t+1)$ where $X(t+1)$ results from using $U(t) = v$. The induction hypothesis implies that

$$\Phi(X(t+1),t+1) \subset \Phi^*(X(t+1),t+1), \tag{83}$$

so that $u \in \Phi^*(X(t+1),t+1)$ and $U(t) = v, U(t+1) = u$ are optimal decisions. We now can use the commutativity of the transitions $p_w$ to show that the decisions $U(t) = u, U(t+1) = v$ have the same expected value and must be optimal too. Thus, $u$ is optimal, contrary to assumption and we must have $u \in \Phi^*(x,t)$.

To complete the proof, we note that starting from $X(t)$, if $X(t+2)$ is the state resulting from $U(t) = v, U(t+1) = u$ and $\tilde{X}(t+2)$ is the state resulting from $U(t) = u, U(t+1) = v$, then commutativity implies that $X(t+2)$ and $\tilde{X}(t+2)$ have the same distribution. By assumption (induction) the decisions $U(t) = v, U(t+1) = u$ are optimal and the value

$$V(X(t),t) = \mathrm{E}\{V(X(t+2),t+2) \mid X(t)\}. \tag{84}$$

Commutativity implies that

$$\mathrm{E}\{V(X(t+2),t+2) \mid X(t)\} = \mathrm{E}\{V(\tilde{X}(t+2),t+2) \mid X(t)\}, \tag{85}$$

which implies that $U(t) = u, U(t+1) = v$ must also be optimal decisions. **Q. E. D.**

**Remark 2.** If $\Phi^*(x,t)$ is the optimal strategy set for $(\mathsf{X}, \mathsf{U}, p_u, R, T)$ as defined in (80), then $\Phi^*$ is necessarily terminal optimal. It also necessarily satisfies the condition for deferrable decisions, simply because the hypothesis of the condition,

$$u \in \Phi^*(x,t), V_v(x,t) > V_u(x,t), \tag{86}$$

is always false. As we indicated in Remark 1, this condition is difficult to check in practice, but we can replace it with stronger conditions which don't refer to the optimal reward function. With these stronger conditions, it's important to have the third condition, commutativity of the transition probabilities, to prove the optimality of a proposed strategy set.

To conclude this section we will prove another stronger condition for deferrable decisions in $\Phi$ based on symmetric MDP problems.

**Definition 5.** The MDP $(\mathsf{X}, \mathsf{U}, p_u, R, T)$ is symmetric if for some $n$

$$\begin{aligned} \mathsf{X} &= \mathbf{X}^n, \\ \mathsf{U} &= \{1,...,n\}, \end{aligned} \tag{87}$$

$$p_{\pi(i)}(\pi y \mid \pi x) = p_i(y \mid x)$$

and

$$R(x) = R(\pi x) \tag{88}$$

where $\pi$ permutes the components of $x, y$, namely

$$\pi x = (x_{\pi(1)},..., x_{\pi(n)}), \tag{89}$$

for any permutation $\pi$ of $\{1,...,n\}$ and all $x \in \mathbf{X}^n$.

**Lemma 1.** Let $V(x,t)$ be the value function for MDP $(\mathsf{X}, \mathsf{U}, p_u, R, T)$ which is symmetric. Then $V(x,t)$ is symmetric in $x_i$ for each $0 \le t \le T$. That is, if $\pi$ is a permutation of $1,...,n$ and $\pi x = (x_{\pi(1)},..., x_{\pi(n)})$, then

$$V(x,t) = V(\pi x, t). \tag{90}$$

In addition,

$$V_i(x,t) = V_{\pi(i)}(\pi x, t). \tag{91}$$

*Proof.* Let $x$ denote a vector in $\mathbf{X}^n = \mathsf{X}$. The dynamic programming equations for the MDP are given as follows. The terminal condition is

$$V(x,T) = R(x). \tag{92}$$

The recursion is

$$V(x,t) = \max_i \{V_i(x,t)\} \tag{93}$$

for times $0 \le t \le T-1$, where

$$V_i(x,t) := \sum_y V(y,t+1)\, p_i(y \mid x). \tag{94}$$

Clearly,

$$V(\pi x, T) = R(\pi x, T) = R(x,T) = V(x,T). \tag{95}$$

Assume that

$$V(x,t+1) = V(\pi x, t+1) \tag{96}$$

for all $x, \pi$ and prove it for $t$. By definition,

$$V_i(x,t) = \sum_y V(y,t+1)\, p_i(y \mid x) \tag{97}$$

and by symmetry assumptions,

$$\begin{aligned} \sum_y V(y,t+1)\, p_i(y \mid x) &= \sum_y V(\pi y, t+1)\, p_{\pi(i)}(\pi y \mid \pi x) \\ &= \sum_y V(y,t+1)\, p_{\pi(i)}(y \mid \pi x). \end{aligned} \tag{98}$$

Thus, it follows that

$$V_i(x,t) = V_{\pi(i)}(\pi x, t). \tag{99}$$

For any permutation $\phi$ and any $y$ it is always true that

$$V(y,t) = \max_i \{V_{\phi(i)}(y,t)\}. \tag{100}$$

In particular, it is true for $\phi = \pi$ and $y = \pi x$. Thus,

$$\begin{aligned} V(\pi x,t) &= \max_i \{V_{\pi(i)}(\pi x,t)\} \\ &= \max_i \{V_i(x,t)\} \\ &= V(x,t). \end{aligned} \tag{101}$$

This completes the induction and the proof. **Q. E. D.**

**Proposition 1.** Suppose that the MDP $(\mathsf{X}, \mathsf{U}, p_u, R, T)$ is symmetric. Then if

$$u \in \Phi(x,t),\ x_v \neq x_u \text{ and } p_v(y \mid x) > 0 \quad \text{imply that} \quad u \in \Phi(y, t+1), \tag{102}$$

then decisions are deferrable in $\Phi$.

*Proof.* Suppose that $u \in \Phi(x,t)$, $V_v(x,t) > V_u(x,t)$ and $p_v(y \mid x) > 0$. Because of the symmetry assumption

$$V_v(x,t) = V_{\pi(v)}(\pi x,t) \tag{103}$$

for all permutations $\pi$. Let $\pi$ be the permutation that interchanges $v$ and $u$. Then if $x_v = x_u$, $V_v(x,t) = V_u(x,t)$. Thus, $V_v(x,t) > V_u(x,t)$ implies that $x_v \neq x_u$. By the proposition's assumption, it follows that $u \in \Phi(y, t+1)$, which proves the result. **Q. E. D.**

## 5.2 APPLICATIONS TO SRM PROBLEMS

Consider the SRM problem where there are $n$ targets and we can only observe one target at a time. In the simplest case, the decision $U(t)$ to make at each time $t$ is only which target $i = 1,...,n$ to observe. There is a Markov chain $X_i(t)$ corresponding to each target $i$, where $X_i(t)$ represents the information state of target $i$ at time $t$. Typically, we assume that the chains $X_i(t)$ are independent and identically distributed, and that the selected (i.e., observed) chain transitions using $p(\xi \mid x)$ and the $n-1$ unobserved chains transition using $q(\xi \mid x)$. Moreover, the reward is typically additive over the $n$ targets, namely

$$R(X_1(T),\ldots,X_n(T)) = \sum_{i=1}^{n} r(X_i(T)). \tag{104}$$

The resulting MDP $(\mathsf{X}, \mathsf{U}, p_u, R, T)$ has special structure where

$$\begin{aligned} &\mathsf{X} = \mathbf{X}^n \text{ and } \mathbf{X} \text{ is the state space of one Markov chain } X_i \\ &\mathsf{U} = \{1,\ldots,n\} \\ &p_i(\xi \mid x) = p_i(\xi \mid x_i)\prod_{j\neq i} q(\xi_j \mid x_j) \text{ for } i \in U, x, \xi \in \mathbf{X}^n \\ &R(x) = \sum_{i=1}^{n} r(x_i) \text{ for } x \in \mathbf{X}^n. \end{aligned} \tag{105}$$

**Remark 3.** If $s = |X|$ is the number of states for each single Markov chain, then the computational complexity of the dynamic programming solution is $O(ns^{2n}T)$. Thus, for fixed $s$ and $T$, the complexity is exponential in $n$. Furthermore, the memory requirements are exponential, namely $O(s^n T)$. In some cases we can find an optimal strategy of the form $U(t) \in \Phi((X_1(t),\ldots,X_n(t)),t)$ where

$$\Phi(x,t) = \left\{ i \,:\, M_i(x_i,t) = \max_j M_j(x_j,t) \right\} \tag{106}$$

This is what we call a priority index rule strategy. The $M_i(x_i,t)$ are indices that can be computed for each target with complexity $O(s^2T)$ (i.e., equivalent to solving the dynamic program for one target). Thus, the complexity of the $n$ target strategy is $O(ns^2T)$ rather than $O(ns^{2n}T)$, linear in $n$ rather than exponential in $n$.

For the class of transition probabilities $p_i(\xi \mid x)$ with structure (105), commutativity is equivalent to the commutativity of the transition functions $p$ and $q$, as the following simple result shows.

**Proposition 2.** If the transition probability functions $p_i(\xi \mid x)$ defined for $\xi, x \in \mathbf{X}^n = \mathsf{X}$ and $i \in \{1,\ldots,n\}$ satisfy

$$p_i(\xi \mid x) = p(\xi_i \mid x_i)\prod_{j\neq i} q(\xi_j \mid x_j), \tag{107}$$

and if for all $\xi_1, x_1 \in \mathbf{X}$,

$$\sum_{\eta_1} q(\xi_1 \mid \eta_1)\, p(\eta_1 \mid x_1) = \sum_{\eta_1} p(\xi_1 \mid \eta_1) q(\eta_1 \mid x_1), \tag{108}$$

then $p_i(\xi \mid x)$ are commutative transition probability functions for $\xi, x \in \mathbf{X}^n = \mathsf{X}$.

*Proof.* Note that for $i \neq j$,

$$\begin{aligned} &\sum_{\eta} p_i(\xi \mid \eta)\, p_j(\eta \mid x) \\ &= \sum_{\eta} p(\xi_i \mid \eta_i) \prod_{k \neq i} q(\xi_k \mid \eta_k)\, p(\eta_j \mid x_j) \prod_{k \neq j} q(\eta_k \mid x_k) \\ &= \sum_{\eta_j} q(\xi_j \mid \eta_j)\, p(\eta_j \mid x_j) \sum_{\eta_i} p(\xi_i \mid \eta_i) q(\eta_i \mid x_i) \prod_{k \neq i,\, j} \sum_{\eta_k} q(\xi_k \mid \eta_k) q(\eta_k \mid x_k). \end{aligned} \tag{109}$$

By assumption

$$\sum_{\eta_j} q(\xi_j \mid \eta_j)\, p(\eta_j \mid x_j) = \sum_{\eta_j} p(\xi_j \mid \eta_j) q(\eta_j \mid x_j) \tag{110}$$

and

$$\sum_{\eta_i} q(\xi_i \mid \eta_i)\, p(\eta_i \mid x_i) = \sum_{\eta_i} p(\xi_i \mid \eta_i) q(\eta_i \mid x_i). \tag{111}$$

Thus,

$$\begin{aligned} &\sum_{\eta_j} q(\xi_j \mid \eta_j)\, p(\eta_j \mid x_j) \sum_{\eta_i} p(\xi_i \mid \eta_i) q(\eta_i \mid x_i) \prod_{k \neq i,\, j} \sum_{\eta_k} q(\xi_k \mid \eta_k) q(\eta_k \mid x_k) \\ &= \sum_{\eta_j} p(\xi_j \mid \eta_j) q(\eta_j \mid x_j) \sum_{\eta_i} q(\xi_i \mid \eta_i)\, p(\eta_i \mid x_i) \prod_{k \neq i,\, j} \sum_{\eta_k} q(\xi_k \mid \eta_k) q(\eta_k \mid x_k) \\ &= \sum_{\eta} p_j(\xi \mid \eta)\, p_i(\eta \mid x), \end{aligned} \tag{112}$$

proving that

$$\sum_{\eta} p_i(\xi \mid \eta)\, p_j(\eta \mid x) = \sum_{\eta} p_j(\xi \mid \eta)\, p_i(\eta \mid x). \tag{113}$$

**Q. E. D.**

**Remark 4.** Note that commutativity always holds if $p$ or $q$ is the identity transition $\delta(\xi_i \mid x_i) = 1$ for $\xi_i = x_i$ and 0 otherwise. Note that $q = \delta$ is assumed true in (non-restless) multi-armed bandit problems. Also, classification sensor management problems often satisfy $q = \delta$ (i.e., the classification information state remains unchanged while the target is unobserved). For this class of MDPs corresponding to SRM problems, the general result (Theorem 1) reduces to the result of Corollary 1.

**Remark 5.** Transition probabilities of the form

$$p_i(\xi \mid x) = p(\xi_i \mid x_i) \prod_{j \neq i} q(\xi_j \mid x_j) \tag{114}$$

and reward functions

$$R(x) = \sum_{i=1}^{n} r(x_i) \tag{115}$$

are obviously symmetric.

**Corollary 1.** Suppose that the MDP $(\mathsf{X}, \mathsf{U}, p_u, R, T)$ has special symmetric structure where

$$\begin{aligned} &\mathsf{X} = \mathbf{X}^n \text{ and } \mathbf{X} \text{ is the state space of one Markov chain } X_i \\ &\mathsf{U} = \{1, \ldots, n\} \\ &p_i(\xi \mid x) = p_i(\xi \mid x_i) \prod_{j \neq i} q(\xi_j \mid x_j) \text{ for } i \in U, x, \xi \in \mathbf{X}^n \\ &R(x) = \sum_{i=1}^{n} r(x_i) \text{ for } x \in \mathbf{X}^n. \end{aligned} \tag{116}$$

Suppose that $\Phi(x,t)$ is a strategy set for $x = (x_1, ..., x_n)$ such that $i \in \Phi(x, T+1)$ implies

$$\sum_{y_i} r(y_i)\, p(y_i \mid x_i) - r(x_i) \geq \sum_{y_i} r(y_j)\, p(y_j \mid x_j) - r(x_j) \tag{117}$$

for all $j \neq i$, and

$$i \in \Phi(x_1, ..., x_i, ..., x_j, ..., x_n, t),\ x_i \neq x_j, \text{ and } p(y_j \mid x_j) > 0 \tag{118}$$

implies that

$$i \in \Phi(x_1,...,x_i,...,y_j,...,x_n,t+1) \tag{119}$$

Then the strategy set $\Phi$ is optimal.

*Proof.* The condition on $p_i(\xi \mid x)$ implies that it is commutative. The second condition implies that $\Phi$ is terminally optimal for the terminal reward $R(x)$, and the third condition implies that decisions in $\Phi$ are deferrable (Proposition 1). The result follows from Theorem 1. **Q. E. D.**

### 5.3 BIRTH-DEATH MDPS

One class of MDPs for which the sufficient conditions hold is analogous to a birth-death process that evolves on an ordered set and can only transition one step at a time. Specifically, suppose that each individual Markov chain $X_i(t)$ has states in the nonnegative integers 0,1,... and can transition down at most one unit at a time (but can transition up any number of units at a time). Thus, $X_i(t+1) \geq X_i(t)-1$. If the Markov chain $X_i(t)$ is in state $0$, it stays there (so that $0$ is a trapping state). The terminal reward gives reward 1 if the state is $0$ and gives reward $0$ for any other state. The objective of the MDP is to control as many of the individual Markov chains into state $0$ by the terminal time $T$ as possible. In this case, the sufficient condition shows that the greedy solution is optimal--that is, select $i$ next for the smallest nonzero state $X_i(t)$.

**Corollary 2.** Suppose that the MDP $(\mathsf{X}, \mathsf{U}, p_u, R, T)$ has the special structure

$$\begin{aligned} &\mathsf{X} = \mathsf{N}_0^n \text{ and } \mathsf{N}_0 = \{0,1,2,\ldots\} \\ &\mathsf{U} = \{1,\ldots,n\} \\ &p_i(\xi \mid x) = p_i(\xi \mid x_i) \prod_{j \neq i} q(\xi_j \mid x_j) \text{ for } i \in U, x, \xi \in \mathsf{N}_0^n \\ &R(x) = \sum_{i=1}^{n} r(x_i) \text{ for } x \in \mathsf{N}_0^n, \end{aligned} \tag{120}$$

where $p(y \mid x) = 0$ for $y < x-1$, $p(0 \mid 0) = 1$, $r(x_i) = 0$ if $x_i \neq 0$, and $r(0) = 1$. For any $x$ and time-to-go $\tau > 0$, define

$$\Phi(x) = \left\{ i : x_i = \min_{x_j > 0} \{x_j\} \right\} \text{ if some } x_j > 0, \tag{121}$$

and

$$\Phi(x) = \{1,\ldots,n\} \text{ if } x_j = 0 \text{ for all } j. \tag{122}$$

Then any control $U(t) \in \Phi(X(t))$ is optimal.

*Proof.* First check that $i \in \Phi(x)$ maximizes the marginal reward

$$\sum_y [r(y) - r(x_i)] p(y \mid x_i) = \begin{cases} 0 \text{ if } x_i > 1 \\ p(0 \mid x_i) \text{ if } x_i = 1. \\ 0 \text{ if } x_i = 0 \end{cases} \tag{123}$$

Suppose $x_i > 0$ and $x_i \le x_j$. If $x_i = 1$, then $i$ has marginal reward $p(0 \mid x_i)$ and $j$ has the same marginal reward $p(0 \mid x_i) = p(0 \mid x_j)$ if $x_i = x_j$, or smaller marginal reward $0$ if $x_j > x_i$. If $x_i \ge 2$, then $i$ and $j$ both have marginal reward $0$. Likewise, if $x_j = 0$ for all $j$, then all $j$ have marginal reward $0$, so that any selection is optimal. Thus, $\Phi$ is terminally optimal.

Suppose that $p(y_j \mid x_j) > 0$ for $x_i \ne x_j$. Then we will show that

$$i \in \Phi(x_1, \ldots, x_n) \text{ implies } i \in \Phi(x_1, \ldots, y_j, \ldots, x_n). \tag{124}$$

Suppose $i \in \Phi(x_1, ..., x_n)$. Note that $x_i > 0$ because $x_i \ne x_j$ and there is at least one non-zero $x_k$. Let $p(y_j \mid x_j) > 0$. If $x_j = 0$, then $y_j = 0$ and none of the $x_k$ change values, and therefore

$$i \in \Phi(x_1, ..., y_j, ..., x_n) = \Phi(x_1, ..., x_j, ..., x_n). \tag{125}$$

If $x_j > 0$, then $x_j > x_i$ since $x_j \ne x_i$. Thus, $p(y_j \mid x_j) > 0$ implies that $y_j \ge x_j - 1$ and $y_j \ge x_i$. Although $y_j$ is non-zero, it is no smaller than $x_i$, and $x_i$ is still the minimum of the non-zero elements of $x_1, ..., y_j, ..., x_n$. Therefore

$$i \in \Phi(x_1, ..., y_j, ..., x_n). \tag{126}$$

It follows that decisions in $\Phi$ are deferrable (Corollary 1), and the proof is complete. **Q. E. D.**

**Example 1.** As an example of the birth-death MDP we have defined, consider the four state model of tracking quality given by the Markov chains shown the figure, showing the possible transitions for the case in which target $i$ is observed or not observed.

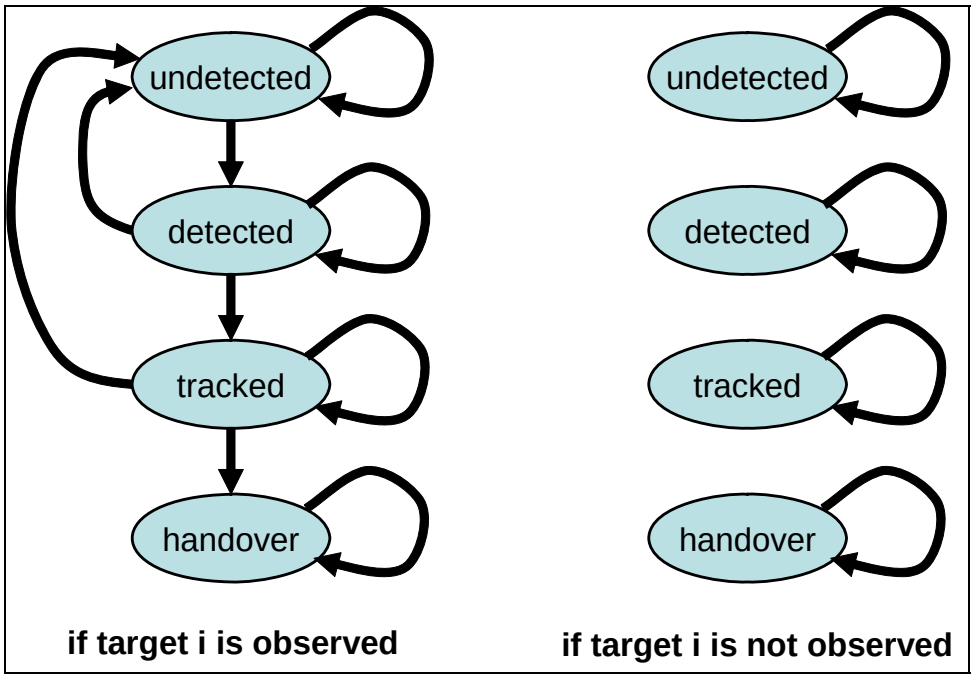


**Figure 19: In example 1, the state of the target remains unchanged if it is not observed, but the state may change, as indicated in the illustration, if the target is observed.**

Here, there are states of tracking accuracy ranging from undetected, detected, tracked, and handover (e.g., to a weapon system). The model requires that the state can transition at most one step to the right at any time interval, but could transition to left all the way to undetected (i.e., drop track). If the track reaches the handover state, it stays there. The goal of the example is to control as many targets into the handover state by time $T$. The transition probabilities reflect the probability of track error increase or decrease, depending on the type of measurements taken. The optimal strategy is to look at the target which is the state closest to but not equal to handover.

## 5.4 BINARY CLASSIFICATION PROBLEM

This problem is to classify as many of $n$ objects over a finite time horizon $T$ given binary measurements of the objects. The problem is interesting because it is a partially observed Markov decision process (POMDP) which can be interpreted as an MDP with a countable state space. Suppose there are $n$ random variables $Z_i$ with values 0, 1 and that $\Pr\{Z_i = 1\} = p$ for all $i = 1,...n$. Suppose that the $Y_i(t)$ are 0,1 observations of $Z_i$, and $Y_i(t)$ are independent and identically distributed conditioned on $Z_i$ with

$$\Pr\{Y_i(t) = y \mid Z_i = z\} = (1-\varepsilon)\cdot\delta_{y,z} + \varepsilon\cdot(1-\delta_{y,z}), \tag{127}$$

where we use the notation $\delta_{y,z} = 1$ if $y = z$ and $0$ otherwise. We assume that $\varepsilon < \frac{1}{2}$. Note that $\varepsilon$ is the probability of classification error for one measurement.

Define the conditional probability $X_i(t)$ as

$$X_i(t) = \Pr\{Z_i = 1 \mid Y_i(1), ..., Y_i(t)\}. \tag{128}$$

The objective of the problem is to maximize the expected reward

$$\mathrm{E}\left\{\sum_{i=1}^{n} r(X_i(T))\right\} \tag{129}$$

at the terminal time $T$, where $r(x_i)$ is the individual reward

$$r(x_i) = \max_{d_i = 0,1}\{r(d_i, 1)x_i + r(d_i, 0)(1 - x_i)\} \tag{130}$$

and $r(d, z)$ are the rewards for the different types of outcomes (i.e., deciding $d_i$ when the true state of $i$ is $z_i$ ).

The processes $X_i(t)$ satisfy

$$X_i(0) = p \tag{131}$$

and for $t \geq 0$,

$$X_i(t+1) = \begin{cases} \dfrac{(1-\varepsilon)X_i(t)}{(1-2\varepsilon)X_i(t) + \varepsilon} & \text{with probability } (1-2\varepsilon)X_i(t) + \varepsilon \\ \dfrac{\varepsilon X_i(t)}{(2\varepsilon - 1)X_i(t) + 1 - \varepsilon} & \text{with probability } (2\varepsilon - 1)X_i(t) + 1 - \varepsilon. \end{cases}. \tag{132}$$

Note that although $X_i(t)$ take values in $\mathsf{R}$, there are only a countable number of possible values they can take. Thus, $X_i(t) \in \mathbf{X} \subset \mathsf{R}$ where $\mathbf{X}$ is a countable set. Thus, we have an MDP $(\mathsf{X}, \mathsf{U}, p_u, R, T)$ where

$$
\begin{aligned}
&\mathsf{X} = \mathbf{X}^n \\
&\mathsf{U} = \{1,\ldots,n\} \\
&p_i(\xi \mid x) = p_i(\xi \mid x_i)\prod_{j\neq i} q(\xi_j \mid x_j) \text{ for } i \in U, x, \xi \in \mathbf{X}^n \\
&R(x) = \sum_{i=1}^{n} r(x_i) \text{ for } x \in \mathbf{X}^n.
\end{aligned}
\tag{133}
$$

where $p(\xi_i \mid x_i)$ is defined by

$$
\begin{aligned}
p\left(\frac{(1-\varepsilon)x_i}{(1-2\varepsilon)x_i+\varepsilon} \mid x_i\right) &= (1-2\varepsilon)x_i + \varepsilon \\
p\left(\frac{\varepsilon x_i}{(2\varepsilon-1)x_i+1-\varepsilon} \mid x_i\right) &= (2\varepsilon-1)x_i + 1 - \varepsilon
\end{aligned}
\tag{134}
$$

and $r(x_i)$ is defined by Equation (130). We will consider the special case for which $r(1,1) = r(0,0) = 1$ and $r(0,1) = r(1,0) = 0$ so that

$$
r(x_i) = \tfrac{1}{2} + \left|x_i - \tfrac{1}{2}\right|, \tag{135}
$$

and we will assume that the prior probability $p = \frac{1}{2}$. Note that if $p = \frac{1}{2}$, then

$$
\mathbf{X} = \left\{ \frac{1}{1+\left(\frac{\varepsilon}{1-\varepsilon}\right)^m} \,:\, m = 0,\pm 1,\pm 2,\ldots \right\}. \tag{136}
$$

**Proposition 3.** The strategy set $\Phi$ defined by

$$
\Phi(x) = \left\{ i \,:\, \left|x_i - \tfrac{1}{2}\right| = \min_j \left|x_j - \tfrac{1}{2}\right| \right\} \tag{137}
$$

is optimal for the binary classification problem with $r(1,1) = r(0,0) = 1$, $r(0,1) = r(1,0) = 0$, and prior probability $p = \frac{1}{2}$ for each object $i$.

*Proof.* The transition probabilities $p_i(\xi \mid x)$ are obviously commutable and symmetric, and the reward function $R(x)$ is obviously symmetric. Note that

$$
\begin{aligned}
&\sum_{y_i}\left[R(y_i)-R(x_i)\right]p(y_i \mid x_i) \\
&= -\tfrac{1}{2}-\left|x_i-\tfrac{1}{2}\right| \\
&\quad+\left(\tfrac{1}{2}+\left|\frac{(1-\varepsilon)x_i}{(1-2\varepsilon)x_i+\varepsilon}-\tfrac{1}{2}\right|\right)\left((1-2\varepsilon)x_i+\varepsilon\right) \\
&\quad+\left(\tfrac{1}{2}+\left|\frac{\varepsilon x_i}{(2\varepsilon-1)x_i+1-\varepsilon}-\tfrac{1}{2}\right|\right)\left((2\varepsilon-1)x_i+1-\varepsilon\right).
\end{aligned} \tag{138}
$$

This simplifies to

$$
\sum_{y_i}\left[R(y_i)-R(x_i)\right]p(y_i \mid x_i) = -\left|x_i-\tfrac{1}{2}\right|+\tfrac{1}{2}\left|x_i-\varepsilon\right|+\tfrac{1}{2}\left|(1-x_i)-\varepsilon\right|, \tag{139}
$$

which is equivalent to

$$
\sum_{y}\left[R(y)-R(x_i)\right]p(y \mid x_i) = \begin{cases} 0 \quad \text{for } 0 \le x_i \le \varepsilon \\ x_i-\varepsilon \quad \text{for } \varepsilon \le x_i \le \tfrac{1}{2} \\ 1-x_i-\varepsilon \quad \text{for } \tfrac{1}{2} \le x_i \le 1-\varepsilon \\ 0 \quad \text{for } 1-\varepsilon \le x_i \le 1 \end{cases}. \tag{140}
$$

Note that because

$$
x_i = \frac{1}{1+\left(\frac{\varepsilon}{1-\varepsilon}\right)^m}, \tag{141}
$$

if $m<0$, then

$$
x_i \le \frac{1}{1+\left(\frac{\varepsilon}{1-\varepsilon}\right)^{-1}} = \varepsilon \tag{142}
$$

and if $m>0$, then

$$
x_i \ge \frac{1}{1+\left(\frac{\varepsilon}{1-\varepsilon}\right)} = 1-\varepsilon. \tag{143}
$$

Thus,

$$\sum_{y_i}\left[R(y_i)-R(x_i)\right]p(y_i\mid x_i)=\begin{cases}0 \text{ for } x_i\neq\frac{1}{2}\\ \frac{1}{2}-\varepsilon \text{ for } x_i=\frac{1}{2}\end{cases}. \tag{144}$$

In particular,

$$\max_j\left\{\sum_{y_j}\left[R\left(y_j\right)-R\left(x_j\right)\right]p\left(y_j\mid x_j\right)\right\}=\begin{cases}0 & \text{if all } x_i\neq\frac{1}{2}\\ \frac{1}{2}-\varepsilon & \text{if some } x_i=\frac{1}{2}\end{cases} \tag{145}$$

and if

$$\left|x_i-\tfrac{1}{2}\right|=\min_j\left|x_j-\tfrac{1}{2}\right|, \tag{146}$$

then

$$\sum_{y_i}\left[R(y_i)-R(x_i)\right]p(y_i\mid x_i)=\max_j\left\{\sum_{y_j}\left[R\left(y_j\right)-R\left(x_j\right)\right]p\left(y_j\mid x_j\right)\right\}. \tag{147}$$

This shows that $\Phi$ is terminally optimal.

To show that decisions in $\Phi$ can be deferred, suppose that $i\in\Phi(x)$ so that

$$\left|x_i-\tfrac{1}{2}\right|=\min_k\left|x_k-\tfrac{1}{2}\right| \tag{148}$$

and suppose $x_j\neq x_i$, $p\left(y_j\mid x_j\right)>0$. Thus,

$$y_j=\frac{(1-\varepsilon)x_j}{(1-2\varepsilon)x_j+\varepsilon} \tag{149}$$

or

$$y_j=\frac{\varepsilon x_j}{(2\varepsilon-1)x_j+1-\varepsilon}. \tag{150}$$

If $\left|x_j - \frac{1}{2}\right| > \left|x_i - \frac{1}{2}\right|$, then it's easy to see that $\left|y_j - \frac{1}{2}\right| \geq \left|x_i - \frac{1}{2}\right|$ and therefore $i \in \Phi\left(x_1,..., y_j,..., x_n\right)$. However, it's possible that $\left|x_j - \frac{1}{2}\right| = \left|x_i - \frac{1}{2}\right|$ and $x_i \neq x_j$. If $x_i \neq \frac{1}{2}$, then the conclusion is not true, because one of the two values of $y_j$ is closer to $\frac{1}{2}$ than $x_i$.

However, we can easily extend the proposition to cover this case. Note that if $\left|x_j - \frac{1}{2}\right| = \left|x_i - \frac{1}{2}\right|$, then $x_j = 1 - x_i$. The classification problem is invariant under the transformation $x_i \to 1 - x_i$, and in particular,

$$V\left(..., x_i,..., \tau\right) = V\left(..., 1 - x_i,..., \tau\right). \tag{151}$$

Furthermore, if $p\left(y_i \mid x_i\right) > 0$, then $p\left(1 - y_i \mid 1 - x_i\right) > 0$. It follows that

$$V_i\left(..., x_i,..., \tau\right) = V_i\left(..., 1 - x_i,..., \tau\right). \tag{152}$$

As a consequence of this and the symmetry of $V$, we find that $\left|x_j - \frac{1}{2}\right| = \left|x_i - \frac{1}{2}\right|$ implies that

$$V_i\left(x, \tau\right) = V_j\left(x, \tau\right). \tag{153}$$

Thus, $i, j \in \Phi\left(x\right)$ implies that $V_i\left(x, \tau\right) = V_j\left(x, \tau\right)$. This is sufficient to extend the proposition because if $V_j\left(x, t\right) > V_i\left(x, t\right)$, then both $x_i \neq x_j$ and $\left|x_j - \frac{1}{2}\right| \neq \left|x_i - \frac{1}{2}\right|$. Thus, we can apply the earlier argument to show that $i \in \Phi\left(x_1,..., y_j,..., x_n\right)$. Consequently, decisions are deferrable in $\Phi$. From Theorem 1 it follows that $\Phi$ is optimal. **Q. E. D.**

## 5.5 SEARCH PROBLEM

Here, we investigate the applicability of the sufficient optimality condition to the search problem considered in [7]. In particular, we consider the search problem where *M* locations are given and each of them may contain an object of interest. We are given a finite-time to search locations and at any time, we can search one location only. Initially, at time *t*=0, for every location *i*, we are given *a priori* probability $x_i(0)$ that location *i* contains an object.

With each location *i*, we associate a hypothesis $H_i$, with $H_i$=1 denoting that there is an object at location *i*. With each location *i*, we also associate a state $x_i(t)$ defined as the probability that object is in location *i*, given the past measurement collection $I(t)$. In other words, $x_i(t)$ is the conditional probability that hypothesis $H_i$ is true:

$$x_i(t) = P\{H_i = 1 \mid I(t)\}. \tag{154}$$

We consider the independent hypothesis assumption, where the events that the hypotheses are true are independent. Under this assumption, the search problem is multi-armed bandit problem. The sufficient optimality condition given in Section 5.1 applies to a sub-class of multi-armed bandit problems. Note, however, that this optimality condition does not apply to the search problem considered in [7] with the exclusive hypothesis assumption. The reason is that, under the exclusive hypothesis assumption, the states of all of the locations are changing after each search, thus violating the basic property of multi-armed bandit problems.

Searching a location results in a measurement $Z$ taking values 0 or 1. The value of the measurement is generated independently at each stage as a random variable with the following probability distribution

$$\begin{aligned} P\{Z=0 \mid H_i=0\} &= P\{Z=1 \mid H_i=1\} = 1-\varepsilon, \\ P\{Z=1 \mid H_i=0\} &= P\{Z=0 \mid H_i=1\} = \varepsilon. \end{aligned} \tag{155}$$

Let $x(t)=(x_1(t),\ldots,x_M(t))$ be the state of locations at time $t$, $u(t)$ be the location searched at time $t$, and $Z(t+1)=z_k$ be the measurement obtained. Then, the new state of locations is given by vector

$$x(t+1) = \left(x_1(t),\ldots, f_i(x_i(t), z_k),\ldots,x_M(t)\right) \qquad \text{with } i = u(t), \tag{156}$$

where $f_i(x_i(t), z_k)$ is a Bayesian update of $x_i(t)$ with new measurement $z_k$:

$$f_i(x_i(t), z_k) = \frac{x_i(t) g(z_k)}{p\left(z_k \mid i, I(t)\right)}, \tag{157}$$

and

$$\begin{aligned} p\left(z_k \mid i, I(t)\right) &= P\{Z(t+1) = z_k \mid u(t) = i, I(t)\} \\ &= x_i(t) g(z_k) + [1 - x_i(t)] f(z_k). \end{aligned} \tag{158}$$

The probability of transitioning from state $x(t)$ to state $\left(x_1(t),\ldots, f_i(x_i(t), z_k),\ldots,x_M(t)\right)$ is equal to $p\left(z_k \mid i, I(t)\right)$, which does not depend on time.

Let $X$ denote the set of all states reachable from the initial state $x(0)$ during the given finite horizon, i.e., the set of all $x$ such that $x$ results from the initial state and some information $I$ that can be collected within the given time horizon.

For the search problem considered in [7], all stage rewards are zero, except for the final-stage reward, which is equal to the probability of the most likely location. In particular,

$$R(x,\tau) = 0 \quad \text{for all } x \in X \text{ and } \tau > 0. \tag{159}$$

$$R(x,0) = \max_k x_k. \tag{160}$$

The goal is find a search strategy that maximizes the final probability of selecting a correct hypothesis, i.e., a strategy $\gamma$ maximizing the expected final-stage reward

$$\underset{\gamma}{E} \max_i x_i(T). \tag{161}$$

The structure of the search problem is identical to the classification example considered in Section 5.4. The only difference is that these two problems have different final-stage rewards. In [7], it is shown that selecting one of the two most likely locations is an optimal search strategy. We will here show this result by using the sufficient optimality condition of Section 5.1.

Let $V(x,\tau)$ be the optimal reward-to-go $\tau$ stages from state $x$, and define functions $V_j$, $j$=1,…,$M$ as follows (see $V_u$ in Equation (75) of Section 5.1):

$$\begin{aligned} V_j(x,\tau) &= \underset{Z}{E} V(x_1,..., f_j(x_j, Z),..., x_M, \tau - 1) \\ &= \sum_{z_k} V(x_1,..., f_j(x_j, z_k)..., x_M, \tau - 1) \cdot p(z_k \mid x_j, j) \end{aligned} \tag{162}$$

for all $x$, $\tau$>0, and all $j$.

For any state $x$ and any stage $\tau$, we define function $\Phi(x,\tau)$ to be the index set of the two most likely locations. In particular, $\Phi(x,\tau)$ is given by

$$\Phi(x,\tau) = \left\{ i \mid x_i = \left[S(x)\right]_1 \text{ or } x_i = \left[S(x)\right]_2 \right\} \text{ for all } x \text{ and } \tau \geq 0., \tag{163}$$

where $S(\cdot)$ is a nonlinear operator from $R^M$ to $R^M$ that maps vector $x$ into a sorted vector $x$, i.e.,

$$S(x) = \left( x_{s(1)},..., x_{s(M)} \right) \quad \text{with} \quad x_{s(1)} \geq ... \geq x_{s(M)}, \tag{164}$$

and $[S(x)]_j$ denotes the $j$-th component of vector $S(x)$.

We next prove that $\Phi$ defines an optimal policy in the sense that for all $x \in X$ and $\tau$ >0,

$$V(x,\tau) = V_i(x,\tau - 1) \text{ for any } i \in \Phi(x,\tau - 1). \tag{165}$$

In particular, we show that Φ satisfies the sufficient optimality conditions given in Proposition 1 of Section 5.1.

**Proposition:** Assume that the initial state is $x(0)=(1/2,\ldots,1/2)$. Then, the function $\Phi(x,\tau)$ has the following properties:

1. For all $x \in X$ , $\tau=0$, and any $i \in \Phi(x,0)$ , we have

$$V_i(x,0) = \max_j V_j(x,0), \tag{165}$$

   where $V_j$ is as defined in (162).

2. For all $x \in X$, $\tau>0$, $i \in \Phi(x,\tau)$, and all $j$ such that $x_i \neq x_j$ and $p(z_k|x_j, u=j)>0$, we have

$$i \in \Phi(x_1,\ldots, f_j(x_j,z_k),\ldots,x_M,\tau-1). \tag{166}$$

***Proof*:** We start by showing that relation (165) holds. The proof is based on the same line of argument as the proof of Proposition 4 in [7]. Note the symmetry assumption in [7] is satisfied [cf. Equation (154)].

Let $x \in X$ be arbitrary, and let $I$ be an information state such that $x$ results from the initial state $x(0)$ and information $I$. By using the definition of $V_j$ [cf. Equation (162)], we have

$$\begin{aligned} V_j(x,0) &= \sum_{z_k} R(x_1,\ldots, f_j(x_j,z_k),\ldots,x_M,0)\, p(z_k \mid j, I) \\ &= \sum_{z_k} \max\left\{x_1,\ldots, f_j(x_j,z_k),\ldots,x_M\right\} p(z_k \mid j, I). \end{aligned} \tag{167}$$

Since reward–to-go $V(x,\tau)$ is invariant under permutations of $x$, [i.e., $V(Px,\tau)=V(x,\tau)$ for any permutation matrix $P$], without loss of generality, we may assume that

$$x_1 \geq x_2 \geq \ldots \geq x_M. \tag{168}$$

Then, we have

$$\max\left\{x_1,\ldots, f_j(x_j,z_k),\ldots,x_M\right\} = \max\{f_1(x_1,z_k), x_2\}, \quad \text{for } j=1 \text{ and all } k, \tag{169}$$

$$\max\left\{x_1,\ldots, f_j(x_j,z_k),\ldots,x_M\right\} = \max\left\{x_1, f_j(x_j,z_k)\right\}, \quad \text{for } j>1 \text{ and all } k. \tag{170}$$

Thus, for $j$=1,

$$
\begin{aligned}
V_1(x,0) &= \sum_{z_k} \max\left\{ f_1(x_1, z_k), x_2 \right\} p(z_k \mid 1, I) \\
&= \sum_{z_k} \max\left\{ \frac{x_1 g(z_k)}{p(z_k \mid 1, I)}, x_2 \right\} p(z_k \mid 1, I) \\
&= \sum_{z_k} \max\left\{ x_1 g(z_k), \quad x_2 p(z_k \mid 1, I) \right\}.
\end{aligned}
\tag{171}
$$

By using the definition of $p(z_k|j,I)$ in Equation (158), we see that

$$
V_1(x,0) = \sum_{z_k} \max\left\{ x_1 g(z_k), \quad x_2 x_1 g(z_k) + x_2 (1 - x_1) f(z_k) \right\}. \tag{172}
$$

Similar to the preceding, for $j \geq 2$, we obtain

$$
\begin{aligned}
V_j(x,0) &= \sum_{z_k} \max\left\{ x_1, f_j(x_j, z_k) \right\} p(z_k \mid j, I) \\
&= \sum_{z_k} \max\left\{ x_1, \frac{x_j g(z_k)}{p(z_k \mid j, I)} \right\} p(z_k \mid j, I) \\
&= \sum_{z_k} \max\left\{ x_1 p(z_k \mid j, I), \quad x_j g(z_k) \right\} \\
&= \sum_{z_k} \max\left\{ x_1 x_j g(z_k) + x_1 (1 - x_j) f(z_k), \ x_j g(z_k) \right\}.
\end{aligned}
\tag{173}
$$

By taking the terms $x_1 x_j f(z_k)$ into a separate summation, we have

$$
V_j(x,0) = \sum_{z_k} \max\left\{ x_1 x_j g(z_k) + x_1 f(z_k), \ x_j g(z_k) + x_1 x_j f(z_k) \right\} - x_1 x_j \sum_{z_k} f(z_k), \tag{174}
$$

By using the relation

$$
\sum_{z_k} f(z_k) = 1 = \sum_{z_k} g(z_k), \tag{175}
$$

we obtain

$$V_j(x,0) = \sum_{z_k} \max\left\{x_1 x_j g(z_k) + x_1 f(z_k),\ x_j g(z_k) + x_1 x_j f(z_k)\right\} - x_1 x_j \sum_{z_k} g(z_k)$$
$$= \sum_{z_k} \max\left\{x_1 f(z_k),\ x_j(1-x_1) g(z_k) + x_1 x_j f(z_k)\right\}. \tag{176}$$

Furthermore, since $x_j \le x_2$ for all $j \ge 2$, we see that

$$V_j(x,0) \le V_2(x,0) \qquad \text{for all} \quad j \ge 2. \tag{177}$$

We now prove that $V_2(x,0)=V_1(x,0)$. For $j=2$, relation (176) gives

$$V_2(x,0) = \sum_{z_k} \max\left\{x_1 f(z_k),\ x_2(1-x_1) g(z_k) + x_1 x_2 f(z_k)\right\}. \tag{178}$$

By changing the variables and by using the symmetry assumption on $f$ and $g$, we obtain

$$V_2(x,0) = \sum_{z_k} \max\left\{x_1 f(b-z_k),\ x_2(1-x_1) g(b-z_k) + x_1 x_2 f(b-z_k)\right\}$$
$$= \sum_{z_k} \max\left\{x_1 g(z_k),\ x_2(1-x_1) f(z_k) + x_1 x_2 g(z_k)\right\}. \tag{179}$$

By comparing this with the expression for $V_1(x,0)$ [cf. Equation (172)], we see that

$$V_2(x,0)=V_1(x,0). \tag{180}$$

Therefore,

$$\max_j V_j(x,0) = V_1(x,0) = V_2(x,0). \tag{181}$$

According to the definition of $\Phi$, we have $\Phi(x,0)=\{1,2\}$, which together with the preceding inequality show that condition shown in Equation (165) holds.

We now show that $\Phi$ satisfies relation in Equation (166). Again, without loss of generality, we may assume that $x_1 \ge x_2 \ge \ldots \ge x_M$, so that $\Phi(x,\tau) =\{1,2\}$. Then, for $i=1$, any $j$ such that $x_j \ne x_1$, and any $z_k$ with $p(z_k| j,I)>0$, we have

$$\text{either} \quad f_j(x_j,z_k) \le x_1 \quad \text{or} \quad f_j(x_j,z_k) > x_1. \tag{182}$$

Hence, $x_1$ is either the best or the second best, implying that

$$1 \in \Phi(x_1, ..., f_j(x_j, z_k), ..., x_M, \tau - 1). \tag{183}$$

Consider now the case where $i$=2 and $j$=1. Similar to the preceding, we have that

$$\text{either} \quad f_1(x_1, z_k) \leq x_2 \quad \text{or} \quad f_1(x_1, z_k) > x_2, \tag{184}$$

implying that $x_2$ is either the best or the second best. Hence, in this case, we have

$$2 \in \Phi(x_1, ..., f_j(x_j, z_k), ..., x_M, \tau - 1), \tag{185}$$

and we are done.

Consider now the case where $i$=2 and $j$>2. We will show that for all $j$>2 with $x_j \neq x_2$ and any $z_k$ with $p(z_k | j, I)$>0, we have

$$x_2 \geq f_j(x_j, z_k). \tag{186}$$

As shown in Section 5.4, the components of $x$ have the following form:

$$x_j = \frac{1}{1 + \left( \frac{\varepsilon}{1 - \varepsilon} \right)^m} \quad \text{for some} \quad m \in \{..., -2, -1, 0, 1, 2, ...\}, \tag{187}$$

where $m$ is the difference between the number of measurements of object $j$ with outcome 1 and the number of measurements of object $j$ with outcome 0. Thus, for some integers $m_2$ and $m_j$, we have

$$x_2 = \frac{1}{1 + \left( \frac{\varepsilon}{1 - \varepsilon} \right)^{m_2}}, \qquad x_j = \frac{1}{1 + \left( \frac{\varepsilon}{1 - \varepsilon} \right)^{m_j}}. \tag{188}$$

Furthermore, since $x_2 \geq x_j$ and $x_j \neq x_2$, from the preceding relation it follows that $m_2 > m_j$. If object $j$ is observed one more time and no measurement is obtained, then the state $x_j$ of the object does not change so $x_2$ is still the second best. If a measurement $z$ is obtained, then the state of object $j$ is given by

$$f_j(x_j, z) = \frac{1}{1+\left(\frac{\varepsilon}{1-\varepsilon}\right)^{m_j-1}} \qquad \text{for } z = 0,$$
$$f_j(x_j, z) = \frac{1}{1+\left(\frac{\varepsilon}{1-\varepsilon}\right)^{m_j+1}} \qquad \text{for } z = 1. \tag{189}$$

Since we have $m_2 \geq m_j+1$, it follows that

$$x_2 = \frac{1}{1+\left(\frac{\varepsilon}{1-\varepsilon}\right)^{m_2}} \geq \frac{1}{1+\left(\frac{\varepsilon}{1-\varepsilon}\right)^{m_j+1}} \geq f_j(x_j, z) \qquad \text{for } z = 0,1. \tag{190}$$

showing that $x_2$ is still the second best. Therefore, $x_2$ remains the second best for any observation outcome, implying that

$$2 \in \Phi(x_1, ..., f_j(x_j, z_k), ..., x_M, \tau - 1), \tag{191}$$

thus showing that $\Phi$ satisfies condition in Equation (166). **Q. E. D.**

# 6 REFERENCES


1. O. P. Kreidl and T. M. Frazier, “Feedback Control Applied to Survivability: A Host-Based Autonomic Defense System”, *IEEE Transactions on Reliability*, 53(1):148-166, 2004.

2. P. Whittle, “Restless bandits: Activity allocation in a changing world”, *Journal of Applied Probability*, 25, 1988.

3. P. Whittle, “Applied probability in Great Britain”, *Operations Research*, 50(1):227--239, 2002.

4. R. B. Washburn, M. K. Schneider, and J. J. Fox, “Stochastic dynamic programming based approaches to sensor resource management”, In Proceedings of *The National Symposium on Sensor and Data Fusion*, 2002.

5. T. M. Cover and J. A. Thomas, “Elements of information theory”, John Wiley & Sons, New York, 1991.

6. Mazor E., Averbuch A., Bar-Shalom Y., and Dayan J, “Interacting multiple model methods in target tracking: a survey”, IEEE Transactions on Aerospace and Electronic Systems, Vol. 34, No. 1, pp.103-123, 1998.

7. D. A. Castañón, “Optimal Search Strategies in Dynamic Hypothesis Testing”, *IEEE Transactions on Systems, Man, and Cybernetics*, 25(7): 1130-1138, 1995.

8. K. Kastella and S. Musick. Comparison of Sensor Management Strategies for Detection and Classification. *Proceedings of the National Symposium on Sensor and Data Fusion*, 1996.

9. R. B. Washburn, A. I. Chao, D. A. Castañón, D. P. Bertsekas, and R. Malhotra, “Stochastic Dynamic Programming for Far-Sighted Sensor Management”, *Proceedings of the National Symposium on Sensor and Data Fusion*, 1997.

10. D. A. Castañón, “Approximate Dynamic Programming for Sensor Management”, *Proceedings of the 36th IEEE Conference on Decision and Control*, San Diego, CA, December 1997.

11. V. Krishnamurthy and R. J. Evans., “Hidden Markov Model Multiarmed Bandits: A Methodology for Beam Scheduling in Multitarget Tracking”, *IEEE Transactions on Signal Processing*, 49: 2893-2907, 2001.

12. R. B. Washburn, M. K. Schneider and J. J. Fox, “Stochastic Dynamic Programming Based Approaches to Sensor Resource Management”, *Proceedings of the 2002 National Symposium and Sensor and Data Fusion*, August 2002.

13. S. Mori, C.-Y. Chong, E. Tse, and R. P. Wishner, “Tracking and Classifying Multiple Targets without A Priori Identification”, *IEEE Transactions on Automatic Control*, AC-31(5): 401-409, 1986.

14. G. Yin and Q. Zhang, “Continuous-Time Markov Chains and Applications: A Singular Perturbation Approach”, Springer, 1997.

15. Y. Kabanov and S. Pergamenshchikov, “Two-Scale Stochastic Systems: Asymptotic Analysis and Control”, Springer, 2002.

# A Lower Bound on Adaptive Sensor Management Performance for Classification

David A. Castañon*
Dept. Electrical & Computer Eng., Boston University
8 St. Mary's St., Boston, MA 02215
email dac@bu.edu

draft

**Abstract**

This paper studies the problem of dynamic adaptive scheduling of multi-mode sensor resources for the problem of classification of multiple unknown objects. Sensor schedules are adapted based on the observed data. The resulting decision problem is formulated as a partially observed Markov decision problem with a large state space. The paper describes a computable lower bound on the achievable performance by a causal adaptive schedule, based on techniques of numerical stochastic control and combinatorial optimization. The lower bound is based on an expansion of the admissible control space of the dynamic decision problem, leading to a problem with simpler decision structure for which the bounds can be computed. The solution of the relaxed problem may be infeasible, but can be used as an approximate scheduling technique in a model predictive control framework. The bound computations are illustrated for several examples involving 100 unknown objects, and compared with the Monte Carlo performance of several sensor scheduling algorithms.

## 1 Introduction

Many modern avionics systems include multiple sensors as well as individual sensors capable of focusing on different objects with different modes. In order to achieve an accurate possible representation of all objects of interest, it is important to coordinate the allocation and scheduling of the different sensors and sensor modes across the different objects of interest. The various modes may be viewed as multiple resources to be managed, and the measurement of different objects under specific modes may be viewed as tasks to be performed with these resources. The adaptive sensor management problem consists of selecting and scheduling the sensor modes which are applied to objects of interest, integrating the collected past information into the selection of future sensing actions.

This paper develops a model for a class of adaptive sensor management problems involving the goal of classifying a known number of objects with unknown type, given a fixed number of sensor resources, where the sensor performance parameters are time-invariant, so that the performance parameters associated with a sensor observing an object with a given mode do not depend on the time that sensing activity occurs. This class of problems arises in several applications, from object classification in surveillance platforms such as Joint STARS, dynamic search, and fault inspection and isolation in manufacturing systems. In these applications, inaccuracies in sensor measurements and variations in object characteristics and pose imply that individual measurements provide noisy estimates of object type whose quality depends on the specific mode used by the sensor. In situations with multiple objects and limited resources, this noisy information can be used to prioritize which objects to look at, and to assign appropriate sensor modes to the objects.

Because of the uncertain nature of the underlying object types and the adaptive nature of the desired schedules, dynamic sensor management problems can be formulated as partially observed Markov decision problems (POMDP) [2, 1, 10, 11]. As such, this class of problems can be solved using stochastic dynamic programming [3]. However, for large numbers of objects, the required state space is very high-dimensional, consisting of the conditional probability

*

distributions of all of the objects. This leads to intractable computational problems, even with the fastest POMDP algorithms.

Sensor management problems have been formulated previously as dynamic optimization problems with partial information. The extensive literature in search theory [20] deals with sensor management problems involving objects that can be of one of two types (hidden or found) with sensors that have only a single mode. The dynamic hypothesis testing problems studied in [6] also have objects that can be of two types and a single sensor mode, but generalize results in search theory to broader classes of measurements. More recently, there has been work [17] using Markov decision problem techniques for sensor management, particularly techniques based on the solution of multiarmed bandit problems. However, these formulations also restrict the sensors to a single sensor with a single mode, and require an infinite horizon, time-invariant formulation.

Because of the complexity of general SM algorithms with multiple sensors and modes, most practical SM algorithms are based on heuristic algorithms based on information-theoretic metrics [5]. To date, there has been no effective approach that can characterize the achievable SM performance to determine whether such heuristic algorithms are performing well.

In this paper, we consider sensor management (SM) problems involving multiple distributed sensors with multiple modes per sensor. This model is an extension of the model discussed in [7]. We show that the resulting POMDP models admit a lower bound based on modifying the constraint structure to expand the space of admissible strategies. The resulting problem becomes a dynamic optimization problem subject to expected value constraints, a class of problems recently studied by Chen and Blankenship in [24]. We develop a hierarchical algorithm that exploits the structure of the resulting relaxed problem. This hierarchical algorithm is based on the solution of single object POMDP problems, coupled with nondifferentiable optimization techniques based on Lagrangian relaxation [16]. The single object problems are of small dimension, and can be readily solved using standard algorithms for POMDPs [10, 11, 13]. The hierarchical algorithm avoids the exponential growth of the dimensions of the resulting state space in the POMDP problem as a function of the number of objects.

The algorithm used to compute the bounds can also be used as a suboptimal algorithm for real-time sensor management. Since the algorithm solves the SM problem with an expanded set of strategies, it is possible that the resulting SM strategies are not feasible. This requires modification of the problem solution, typically using a receding horizon technique similar to model-predictive control. We describe in the paper one such approach at this SM algorithm. The paper includes several examples where the lower bound performance is computed, and compared with the Monte Carlo performance achieved by suboptimal SM algorithms.

The remainder of this paper is organized as follows. Section II includes the mathematical statement of the sensor management problem for a single sensor, and discusses the stochastic dynamic programming algorithm for this problem. Section III describes the modified SM formulation and the derivation of the lower bound. Section IV describes computation approaches for evaluating the lower bound. Section V discusses extensions of the earlier results to multiple sensors. Section VI describes the numerical experiments and results. Section VII is a summary of the results and discussion of open problems of interest.

# 2 Problem formulation

In this section, we develop a formulation of the adaptive SM problem as a partially observed Markov decision problem (POMDP). Assume that there are $N$ objects of interest in the problem. Each object can belong to one and only one of $K$ different classes, and the object identity does not change over time. Let the variable $x_i \in X \equiv \{1, \ldots, K\}$ denote the true class of object $i$. We define the complete (but unknown) system state as:

$$\underline{x} = \begin{pmatrix} x_1 & x_2 & \cdots & x_N \end{pmatrix} \tag{1}$$

Since the identities do not change over time, the complete system state is constant over time. We assume that $x_i$ are independent random variables with values in the finite space $X$. Associated with each object $i$ is a prior probability vector $\pi_i(0)$ which describes the probability distribution of the random variable $x_i$. That is,

$$\pi_{ij}(0) = \text{Prob}\{x_i = j\} \tag{2}$$

These probability distributions represent a priori knowledge collected on each object before the start of the SM problem.

In order to obtain information about the state of each object, selected objects are examined with different modes from different sensors. In order to simplify the notation in the exposition, we consider the case of a single sensor with multiple modes $m \in \{1, \ldots, M\}$. We will highlight later the extensions required to incorporate multiple sensors. The action to use a sensor mode $m$ on object $i$ produces an observable $y_m$ in a finite set $Y_m$, with a conditional probability distribution that depends only on the object $i$, its type $x_i$ and the mode $m$, denoted by $p(y_m|i, x_i, m)$. We assume that the observation outcomes of these sensing actions are conditionally independent of each other given the object types.

We assume that obtaining a measurement of object $i$ with mode $m$ requires sensor resources $R_{im} > 0$ (e.g. duty cycle of a radar), which depend on the specific object and mode selected. For the SM problem of interest, the sensor has a finite amount of sensor resources $R$ that can be used for measuring objects. The objective is to classify, with minimal error cost, the objects after the sensor resource $R$ is exhausted. This formulation is stated more rigorously below.

Without loss of generality, we restrict our attention to SM strategies that execute only one action at a time. Such strategies are optimal in that they provide maximal information for adaptation, and will achieve minimal error cost. Let $u(k) = (i(k), m(k))$ denote the $k+1$-th action (starting at $k = 0$) taken by the sensor, consisting of measuring object $i(k)$ with mode $m(k)$. Let $U$ denote the set of possible sensor actions, and let $y_{m(k)}(k)$ denote the measured value resulting from action $u(k) \in U$. The past information available to adaptively select $u(k)$ is $I(k) = \{u(0), y_{m(0)}(0), \ldots, u(k-1), y_{m(k-1)}(k-1)\}$. The SM problem decisions are selected adaptively until a final random stopping instance $T$, selected based on the information $I(T)$. At the end of this stopping instance, the information $I(T)$ is available for estimating the object types. For each object $i$, there is a final decision $v_i \in X$ based on $I(T)$ that is selected to minimize the expected classification error.

An admissible adaptive SM policy is a set of measurable feedback strategies $\{\gamma(0), \ldots, \gamma(T)\}$ and stopping time $T$ such that

$$\begin{aligned} &\gamma(k) : I(k) \to U, \ k < T \\ &T : I(T) \to \{stop, continue\} \\ &\gamma(T) : I(T) \to X^N \end{aligned} \tag{3}$$

Let $\Gamma$ denote the set of all admissible SM policies. Since the observation space is finite and the decision space is also finite, $\Gamma$ is a countable space.

Denote by $c(v, x)$ the cost of selecting classification decision $v$ when the true object type is $x$. The SM problem statement is to minimize the expected total classification cost

$$J(\gamma) = E_\gamma\{\sum_{i=1}^{M} c(v_i, x_i)\} \tag{4}$$

over adaptive SM policies $\gamma \in \Gamma$ satisfying the resource utilization constraint

$$\sum_{k=0}^{T-1} R(u(k)) \leq R \tag{5}$$

with the notation $R(u(k)) \equiv R_{i(k)m(k)}$. Note that the constraint in (5) is a sample path constraint; for every realization of the information sets $I(k)$, the adaptive policy $\gamma$ must not exceed the total sensor resources available. Note also that, given the finite state nature of the set of possible observation outcomes per mode $Y_m$ and possible decisions $u_m$, the number of possible information sets after $k-1$ actions $I(k)$ is countable. This implies that there is a finite number of possible admissible SM policies that satisfy the constraint (5).

The above problem is a class of finite-state, finite-observation partially observed Markov decision problems studied in [2, 1, 11, 10, 3], with the special structure that the underlying state dynamics are trivial, and the presence of the sample path constraints of (5). Such problem scan be transformed into fully-observed Markovian decision problems in terms of a sufficient statistic: the conditional probability distribution of the state $\underline{x}$ given information $I(k)$, as follows: Let $S \subset R^K$ denote the space of probability distributions on $X$, and let $S_N$ denote the space of probability distributions on $X^N$. The conditional distribution vector for the composite state $\underline{x}$ given the information $I(k)$, $P(\underline{x}|I(k)) \in S_N$, can be viewed as an information state, a sufficient statistic summarizing the

past observations. The recursive evolution of this information state in response to an action $u(k) = (i(k), m(k))$ can be described by Bayes' rule as

$$\begin{aligned} P(\underline{x}|I(k+1)) &= P(\underline{x}|I(k), u(k), y_{m(k)}(k)) && (6)\\ &= \frac{P(y_{m(k)}(k)|\underline{x}, I(k), u(k))P(\underline{x}|I(k))}{P(y_{m(k)}(k))|I(k), u(k))} && (7)\\ &= \frac{P(y_{m(k)}(k)|x_{i(k)}, m(k))P(\underline{x}|I(k))}{P(y_{m(k)}(k))|I(k), u(k))} && (8) \end{aligned}$$

with the initial condition

$$P(\underline{x}|I(0)) = \prod_{i=1}^{N} \pi_i(0) \tag{9}$$

Under the previous independence assumptions, the following lemma establishes a convenient representation:

**Lemma 2.1** *Under the SM problem assumptions, the conditional probability*

$$P(\underline{x}|I(k)) = \prod_{i=1}^{N} P(x_i|I(k)) \tag{10}$$

*where the evolution of $P(x_i|I(k))$ under sensing action $u(k) = (i(k), m(k))$ and observed value $y_{m(k)}(k)$ is given by*

$$P(x_i|I(k+1)) = \begin{cases} P(x_i|I(k)) & \text{if } i(k) \neq i \\ \frac{P(y_{m(k)}(k)|x_{i(k)}, m(k))P(x_i|I(k))}{\sum_{j=1}^{K} P(y_{m(k)}(k)|x_i=j, I(k))P(x_i=j|I(k))} & \text{otherwise} \end{cases} \tag{11}$$

The proof of this lemma is straightforward by induction, as the independence assumption of the object types $x_i$ guarantees the Lemma is satisfied at $k = 0$, and (8) establishes the recursion. Note also that $P(x_i|I(k))$ depends only on measurements in $I(k)$ corresponding to object $i$.

The importance of Lemma 2.1 is that we can characterize the information state as a product of marginal distributions, in $S^N$, as opposed to a joint distribution in $S_N$. As notation, define $\pi_i(k)$ to be the conditional probability distribution of $x_i$ given information $I(k)$:

$$\pi_i(k) = P(x_i|I(k)) \tag{12}$$

The vector $\pi_i(k)$ has components $\pi_{ij}(k) = P(x_i = j|I(k))$. The results of Lemma 2.1 establish the following representation for the conditional probability distribution of the entire state: $P(\underline{x}|I(k))$ can be computed from $\pi_i(k), i = 1, \ldots, N$. Define the information vector

$$\vec{\pi} = \begin{pmatrix} \pi_1 \\ \vdots \\ \pi_N \end{pmatrix} \tag{13}$$

For a given observation $y_m$ using mode $m$ on object index $i$, define the observation probability matrix as the $K \times K$ diagonal matrix

$$B_i(y_m) = \text{diag}\{P(y_m|x_i = 1, m), P(y_m|x_i = 2, m), \ldots, P(y_m|x_i = K, m)\} \tag{14}$$

With this notation, we can define the evolution of the information vector in response to a measurement $y_m$ obtained from a sensing action $(i, m)$ in terms of an evolution operator on $(S^N, U, Y)$ as

$$T(\vec{\pi}, u = (i, m), y) = \begin{pmatrix} \pi_1 \\ \vdots \\ \pi_{i-1} \\ \frac{B_i(y)\pi_i}{e^T B_i(y)\pi_i} \\ \pi i + 1 \\ \vdots \\ \pi_N \end{pmatrix} \tag{15}$$

where $e$ is a $K$-dimensional vector of all ones.

Conceptually, the SM problem described above can be solved by stochastic dynamic programming [3]. The resource constraint in (5) can be incorporated into the dynamics to obtain a dynamic programming recursion, as follows. Define a value function $V(\vec{\pi}, C)$ to be the optimal solution of SM in (3)-(5) when the initial information is $\vec{\pi}$ and the available sensor resource level is $R = C$. The value function $V$ is thus defined on $S^N \times R^+$. The SM optimization problem is stated as a total cost problem with nonnegative costs, for which the optimal value function satisfies Bellman's equation [3], as described below. Let $U(R) \subset U$ denote the set of sensor actions $(i, m)$ such that $R_{im} \leq R$; this is the subset of sensor actions that are feasible when there are only $R$ resources left. At each decision stage, there is a choice of stopping and classifying the objects with the available information, or taking additional measurements. The optimal value function therefore satisfies the Bellman equation

$$\begin{aligned} V(\vec{\pi}, R) &= \min\Big[\sum_{i=1}^{N} \min_{v_i \in X} \sum_{j=1,\ldots,K} c(v_i, j)\pi_{ij}, \\ &\qquad \min_{u \equiv (i',m') \in U(R)} E_y\{V(T(\vec{\pi}, u, y), R - R_{i'm'})\}\Big] \end{aligned} \tag{16}$$

where

$$E_y\{V(T(\vec{\pi}, u, y), R - R_{i'm'})\} = \sum_{y \in Y_{m'}} P(y|I(k), u) V(T(\vec{\pi}, u, y), R - R_{i'm'}) \tag{17}$$

$$= \sum_{y \in Y_{m'}} e^T B_{i'}(y) \pi_i V(T(\vec{\pi}, u, y), R - R_{i'm'}) \tag{18}$$

This recursion defines the optimal value function from a given information vector and a given resource level in terms of the value function at other information vectors evaluated with strictly less resource levels. Furthermore, we have boundary conditions for this recursion as follows: Let $R_{min} = \min_{i,m} R_{im}$. Then, the set of admissible modes $U(R)$ is empty for $R < R_{min}$. Thus,

$$V(\vec{\pi}, R) = \sum_{i=1}^{N} \min_{v_i \in X} \sum_{j=1,\ldots,K} c(v_i, j)\pi_{ij} \quad \text{if } R < R_{min} \tag{19}$$

Eqs. (16)-(19) can be used recursively to compute the optimal value for all information states and nonnegative resource levels.

Note that the initialization of the recursion decouples into $N$ independent optimizations, as there are no coupling constraints on the decisions $v_i$, and the local decision costs $c(v_i, x_i)$ depend only on the marginal probability distributions of each object's type. However, the recursion (16) does not preserve this decomposability. The coupling arises primarily because of the resource use constraints in (5); the decision of which object to view and which mode to use depends on the information vector of all the objects and the available resources. Thus, the dynamic programming induction must be carried out for the entire state $\vec{\pi}(t)$, which becomes a formidable computation problem even for moderate numbers of objects.

# 3 Relaxed Formulation and Lower Bounds on Classification Performance

A possible approach to overcoming the computational difficulty of the previous formulation is to relax the sample path sensor resource use constraints (5) and use an averaged version of the same constraints, as

$$E\{\sum_{k=1}^{T} R(u(k))\} \leq R \tag{20}$$

This approach replaces a large set of constraints (one per sample path) by a single aggregate constraint. Note that any SM strategies that satisfy (5) will also satisfy (20). Thus, this approach increases the set of admissible SM strategies. Let $J^*$ denote the optimal classification cost of the original SM problem in (4)-(3) with constraints (5). Let $J_A^*$ denote the optimal classification cost of the SM problem in (4)-(3) with constraints (20). This leads to the following lemma:

**Lemma 3.1** $J^* \geq J_A^*$

The relaxed SM problem has a single coupling constraint relating the sensing actions on different objects. This structure can be exploited using Lagrange multipliers as follows. Let $\lambda \geq 0$ denote a Lagrange multiplier. Consider the new SM objective for admissible SM policies in $\Gamma$ as

$$J(\lambda, \gamma) = E_\gamma\{\sum_{i=1}^{N} c(v_i, x_i)\} + \lambda[E_\gamma\{\sum_{k=0}^{T-1} R(u(k))\} - R] \tag{21}$$

Consider now the unconstrained SM problem of finding adaptive SM strategies $\gamma$ and an adaptive stopping time $T$ to minimize (21). If $(\gamma, T)$ is an adaptive SM policy that satisfies (20), the second term in (21) is nonpositive. Denote by $J^*(\lambda)$ the optimal value of (21) over all adaptive SM strategies $\gamma \in \Gamma$. Then,

**Lemma 3.2** *For all values of* $\lambda \geq 0$,

$$J^* \geq J_A^* \geq J^*(\lambda) \tag{22}$$

*In particular,*

$$J^* \geq \sup_{\lambda \geq 0} J^*(\lambda) \tag{23}$$

Lemma 3.2 is a consequence of weak duality in nonlinear programming [4]. Note that the number of adaptive SM strategies that satisfy (21) is finite, because the set of possible histories $I(k)$ is finite for all $k$. Thus, computation of $J_A^*$ is an integer programming problem, and computation of $\sup_{\lambda \geq 0} J^*(\lambda)$ is its dual problem. The key issue is whether the lower bounds $J^*(\lambda)$ can be computed efficiently. Rewrite (21) for $\gamma \in \Gamma$ as

$$J(\lambda, \gamma) = E_\gamma\{\sum_{i=1}^{N}[c(v_i, x_i) + \lambda \sum_{k=0}^{T-1} R(u(k))\delta(i(k) - i)]\} - \lambda R \tag{24}$$

where the indicator function $\delta(i) = 1$ if $i = 0$, and 0 otherwise. This suggests that optimization of $J(\lambda)$ may be separable across individual objects $i$.

Partition the information $I(k)$ into disjoint sets $I_i(k)$, where $I_i(k)$ are the sensing actions and measurement actions applied to object $i$:

$$I_i(k) = \{(u(j), y(j)) | j < k, i(j) = i\} \tag{25}$$

Note that the conditional probability vector $\pi_i$ only changes on measurements included in $I_i(k)$. We wish to restrict the set of adaptive SM strategies to a subset where the decision to apply a sensor action for object $i$ depends only on the information previously collected for object $i$. We refer to this subset of strategies as adaptive *local* SM strategies, defined as:

**Definition 3.1** *An adaptive* local *SM policy is an adaptive SM policy* $\gamma$ *and stopping times* $T_i, i = 1, \ldots, N$, *with the properties that, for each sensing action instance* $k$,

1. *If* $u(k) = (i(k), m(k))$, *then* $i(k) = k \mod N + 1$.
2. *The selected sensor mode* $m(k)$ *depends only on the information* $I_{i(k)}$.
3. *For each object* $i$, *there is a stopping time* $T_i$ *which depends only on* $I_i(T_i)$ *such that, for all* $k \geq T_i$, *if* $i = k \mod N + 1$, *no sensing action is taken. If* $k < T_i$ *and* $i = k \mod N + 1$, *then* $u(k) = (i, m)$ *for some mode* $m$ *in* $\{1, \ldots, M\}$.
4. *At time* $T_i$, *the local decision* $v_i$ *for object* $i$ *is selected as a function of* $I_i(T_i)$.

Adaptive local SM strategies use a round-robin schedule for selecting which objects to measure. Thus, the choice of sensing object for each action is not adapted to the prior information. Furthermore, the choice of sensing mode for each action on object $i$ depends only on the prior information collected on that object. In addition, there is an independent stopping time for each object $i$ such that a final classification decision is made on object $i$, based only on prior information collected on that object. Note that there are decision instances $k$ where no sensing action

is taken, when $k \geq T_i$ and $i = k \mod N + 1$; these instances correspond to times after a final decision has been selected for object $i$. The *effective* stopping time of an adaptive local SM policy is defined as $T = \max_{i=1,\ldots,N} T_i$, and is the earliest time at which every object has a final classification decision. Thus, adaptive local SM strategies can be viewed as a subset of the class of adaptive SM strategies.

Let $\Gamma_L$ denote the set of adaptive local SM policies. For a given amount of sensor resources $R$, there are a finite number of feasible adaptive local SM strategies. In general, $\Gamma_L$ is a countable discrete set. For the purposes of bound computation, we will expand $\Gamma_L$ to include *mixed* policies, consisting of probabilistic mixtures of policies in $\Gamma_L$:

**Definition 3.2** *A mixed local SM policy is a probability distribution $q(\gamma)$ over $\Gamma_L$ such that local SM policy $\gamma$ is selected for use with probability $p(\gamma)$. The set of mixed local SM strategies is denoted by $Q(\Gamma_s)$.*

Consider the problem of minimizing the relaxed cost (24) over local SM policies $\Gamma_L$. Since $\Gamma_L \subset \Gamma$, we have

$$\min_{\gamma \in \Gamma} J(\lambda, \gamma) \leq \min_{\gamma \in \Gamma_L} J(\lambda, \gamma) \tag{26}$$

Furthermore, since (24) is an unconstrained objective, the minimum in mixed local SM policies is achieved by a pure local SM policy, so

$$\min_{\gamma \in \Gamma} J(\lambda, \gamma) \leq \min_{q \in Q(\Gamma_L)} \sum_{\gamma \in \Gamma_L} q(\gamma) J(\lambda, \gamma) \tag{27}$$

The importance of mixed local SM strategies is highlighted in the theorem below.

**Theorem 3.1** *Consider any admissible adaptive SM policy $\gamma \in \Gamma$. Then, there exists a mixed local SM policy $q \in Q(\Gamma_s)$ such that the expected classification costs in (4) and the expected total resource use in (20) are equal under both policies $\gamma$ and $q$.*

The proof of this result is by construction, and is included in the Appendix. This result implies the following inequality:

$$\min_{\gamma \in \Gamma} J(\lambda, \gamma) \geq \min_{q \in Q(\Gamma_L)} \sum_{\gamma \in \Gamma_L} q(\gamma) J(\lambda, \gamma) \tag{28}$$

Combining (27) and (28) yields the following:

$$\min_{\gamma \in \Gamma} J(\lambda, \gamma) = \min_{q \in Q(\Gamma_L)} \sum_{\gamma \in \Gamma_L} q(\gamma) J(\lambda, \gamma) = \min_{\gamma \in \Gamma_L} J(\lambda, \gamma) \tag{29}$$

Eq. (29) implies that lower bounds for the achievable classification performance can be computed by optimizing over local SM policies only. For each local SM policy $\gamma \in \Gamma_L$, let $\gamma_i$ denote the policy that is used for instances $k$ when actions are taken for object $i$, and let $\Gamma_{Li}$ be the set of such admissible local SM policies for object $i$. Thus, $\gamma_i$ selects actions for object $i$ based on past observations $I_i(k)$, and selects a stopping time $T_i$ and a final classification $v_i$ at that stopping time. The importance of local SM policies is that the optimization in (29) decouples over objects as

$$\begin{aligned} \min_{\gamma \in \Gamma_L} J(\lambda, \gamma) &= E_\gamma \{ \sum_{i=1}^{N} [c(v_i, x_i) + \lambda \sum_{k=0}^{T-1} R(u(k)) \delta(i(k) - i)] \} - \lambda R \\ &= \sum_{i=1}^{N} \min_{\gamma_i} E_{\gamma_i} [c(v_i, x_i) + \lambda \sum_{k \geq 0 : i = k \mod N+1}^{T_i - 1} R(u(k))] - \lambda R \end{aligned}$$

This implies that computation of the bounds can be achieved with $N$ independent optimization problems for each value of $\lambda$. Furthermore, the optimal bound can be computed as in Lemma 3.2, as

$$J^* \geq \sup_{\lambda \geq 0} \{ \sum_{i=1}^{N} \min_{\gamma_i} E_{\gamma_i} [c(v_i, x_i) + \lambda \sum_{k \geq 0 : i = k \mod N+1}^{T_i - 1} R(u(k))] - \lambda R \} \tag{30}$$

Note that the right hand side of (30) is the dual of the following linear programming problem:

$$\min_{q \in Q(\Gamma_L)} \sum_{\gamma \in \Gamma_L} q(\gamma) E_\gamma J(\gamma) \tag{31}$$

subject to

$$\sum_{\gamma \in \Gamma_L} q(\gamma) E_\gamma [\sum_{k=0^{T-1}} R(u(k))] \leq R \tag{32}$$

$$\sum_{\gamma \in \Gamma_L} q(\gamma) = 1 \tag{33}$$

which is a linear program over the choice of probability distributions $q \in Q(\Gamma_L)$. This can be exploited to solve efficiently for the bound. Specifically, note that this is a linear program subject to two constraints, which implies that the optimal mixed local SM policy $q$ will have support only on two pure local SM policies. This property will be exploited in the next section for bound computation.

# 4 Computation of the Lower Bound

There are two potential approaches to compute a lower bound: a dual approach, based on Lagrangian relaxation [16], that optimizes (30) over the choice of dual variable $\lambda$, and a primal approach based on solving the linear program (31)-(33). The dual approach is straightforward, and uses techniques from nondifferentiable optimization [19] to search the space of possible $\lambda$. The primal approach is harder, because the optimization is over a large space of possible values of mixture probabilities $q$. However, this mixture has very sparse support, which makes it suitable for column generation algorithms [18].

A fundamental step in either approach is the computation of the optimal local SM strategies for a fixed value of $\lambda$ for each object. For object $i$, one must solve the local problem given $\lambda$:

$$\min_{\gamma_i} E_{\gamma_i}[c(v_i, x_i) + \lambda \sum_{k \geq 0 : i = k \mod N+1}^{T_i - 1} R(u(k))] \tag{34}$$

This problem is a multi-stage single object partially observed Markov decision problem, with sufficient statistic given by the marginal probability distribution $\pi_i(k)$. Furthermore, we can reduce the action instants to a new counter $k'$ indexing only the action opportunities for object $i$, to obtain

$$\min_{\gamma_i \in \Gamma_{Li}} E_{\gamma_i}[c(v_i, x_i) + \lambda \sum_{k'}^{T_i' - 1} R_{im(k')}] \tag{35}$$

The resulting POMDP problems are small enough to solve using existing algorithms such as those overviewed in [1, 11, 10, 13, 14]. These algorithms exploit Smallwood and Sondik's efficient parameterization [2] of the optimal cost-to-go at stage $k'$ as a minimum of linear functions of the statistic $\pi_i(k')$, and are efficient for problems with a few discrete true states.

Solution of the $N$ decoupled problems (35) yields a local SM policy $\gamma \in \Gamma_L$, for which the expected classification cost $E_\gamma[\sum_{i=1}^N c(v_i, x_i)]$ and expected resource use $E_\gamma[\sum_{k=0}^{T-1} R(u(k))]$ are computed from the solution. This provides the starting point for the use of column generation [18] for solution of (31)-(33). Column generation was used by Yost [21, 22, 23] in his work on POMDPs for resource assignment and was also exploited in [8] for the solution of stochastic weapon assignment problems. The main result of [21, 22, 23] is an efficient constraint generation algorithm which solves the linear program in (31)-(33) while considering only mixtures of a very small number of local strategies. We summarize their algorithm below.

The algorithm starts with an initial set of pure local SM policies $\gamma^d$ indexed by $d = 1, \ldots, D$, with known expected classification performance $J^d$ and expected resource use $R^d$. The first step in the algorithm is to solve the linear program in (31)-(33) restricted to mixtures of the $d = 1, \ldots, D$ initial policies. Since the support of the admissible mixed policies is restricted, the solution provides an upper bound $J^{UB}$ to the optimal cost. Denote by

$\lambda_D$ the optimal dual price of the resource constraint (33) in this solution. The constraint generation algorithm uses this optimal dual price value in (35) to generate a new candidate local SM policy $\gamma^{D+1}$, solving $N$ independent POMDP problems. The combined solution of the $N$ subproblems also provides a lower bound $J^{LB}$ on the optimal performance, as described in Lemma 3.2. The key result in the constraint generation algorithm is stated as follows

**Lemma 4.1** *Consider the pure local SM policy generated by the solution of (35). If $J^{LB} = J^{UB}$, the optimal solution over all mixtures of local SM policies is a mixture of the local strategies indexed by $d = 1, \ldots, D$. Otherwise, the pure local SM policy $\gamma^{D+1}$ can be used as part of a mixed strategy which provides a cost lower than $J^{UB}$.*

The proof of this result is given by Gilmore and Gomory [18]. It is based on the fact that solving the decoupled dual problem (35) is equivalent to finding the local SM policy which has the greatest impact in reducing the cost of the current best mixture. This leads to a dynamic column generation algorithm, as follows: if $J^{LB} < J^{UB}$, increase the number of local SM policies considered in the LP by adding the new pure local SM policy $\gamma^{D+1}$, and resolve the primal problem in (31-33) with support restricted on $\{\gamma^1, \ldots, \gamma^{D+1}\}$. For the optimal dual value, solve the relaxed problem in (35), and compare the new upper and lower bounds. Each iteration, reduces the upper bound, until the lower bound and upper bound estimates are close enough. By the lemma above, the optimal solution will be obtained without enumerating all of the pure local strategies.

# 5 Extension to Multiple Sensors

The development of the previous sections carries through with little modification when multiple sensors are used. The key difference is that there is a separate resource constraint for each sensor. Thus, there will be a vector of sensor resources $R_s$, where $s$ is a sensor index, thus resulting in a vector of averaged constraints (20). The Lagrange multipliers $\lambda$ will thus be vectors instead of scalars. Nevertheless, all of the lemmas and theorems can be extended to the multisensor case with minor modifications.

The main assumption that was used in the single sensor formulation was that only one sensor action would be performed simultaneously. While this assumption is accurate for single sensor problems, it is an optimistic assumption for multiple sensor problems where time or duty cycle is the main resource. Multisensor problems are often required to operate the sensors simultaneously, thereby potentially degrading the achievable performance. However, note that the local SM strategies that are used in the lower bound computation allow for the parallel execution of sensing actions on different objects, so as long ans the number of objects is greater than the number of sensors, there won't be much performance degradation from executing simultaneous sensing actions.

The column generation algorithm discussed in the previous section extends naturally to multiple sensors. When there are $L$ sensors, the optimal mixed local SM policies will be mixtures of $L + 1$ pure local SM policies. Nondifferentiable optimization algorithms that maximize the dual cost can also be used in this case.

# 6 Examples

In this section, we present computational experiments comparing the lower bounds described in the previous section with the Monte Carlo performance of a pair of SM feedback policies.

We consider scenarios involving a single sensor with 100 unknown objects. The objects can be of three different types ($K = 3$), corresponding to cars, trucks and military vehicles. The sensor can be electronically steered to collect images of each object; the sensor has a low resolution mode that takes 1 second per image ($R_{i1} = 1$), and a higher resolution mode that requires 5 seconds per image, ($R_{i2} = 5$). Low resolution imagery is useful in separating cars from trucks and military vehicles, but separating trucks from military vehicles requires high resolution imagery.

In the experiments, we start with the a priori information that there are on average 10 military vehicles, 20 trucks and 70 cars in a group of 100 objects. Thus, each object has an initial probability distribution over type of $(0.1, 0.2, 0.7)$, where types are indexed as military vehicle, truck and car. We assume that the images generated by the sensor are processed into binary outputs, where $y_{ij} = 1$ indicates that object $i$ is estimated to be potentially a military vehicle, and $y_{ij} = 2$ indicates that object $i$ is likely not to be a military vehicle.

The objective of the problem is to determine as accurately as possible which objects are military vehicles (type 1). Thus, the classification costs are given by $d(v_i, x_i)$ as a $3 \times 3$ matrix where $v_i$ is the row index:

$$(d(v_i, x_i)) = \begin{pmatrix} 0 & MD & MD \\ FA & 0 & 0 \\ FA & 0 & 0 \end{pmatrix} \tag{36}$$

where $MD, FA$ will be variables in the experiments representing false alarm and missed detection costs. In the experiments, FA is kept constant to 1, while MD varies from 1 to 80, indicating the relative cost of failing to classify correctly a military vehicle.

To complete the problem specification, we need to describe the conditional probability distribution of the measurements and the constraints on the decisions. The conditional probability distributions $p(y|x, m)$ are given by:

$$\begin{aligned}
p(y_1 = 1|1, 1) = 0.90 \quad &; \quad p(y_1 = 2|1, 1) = 0.10 \\
p(y_1 = 1|2, 1) = 0.90 \quad &; \quad p(y_1 = 2|2, 1) = 0.10 \\
p(y_1 = 1|3, 1) = 0.10 \quad &; \quad p(y_1 = 2|3, 1) = 0.90 \\
p(y_2 = 1|1, 2) = 0.80 \quad &; \quad p(y_2 = 2|1, 2) = 0.20 \\
p(y_2 = 1|2, 2) = 0.15 \quad &; \quad p(y_2 = 2|2, 2) = 0.85 \\
p(y_2 = 1|3, 2) = 0.05 \quad &; \quad p(y_2 = 2|3, 2) = 0.95
\end{aligned}$$

Note that mode 1 is unable to distinguish between types 1 and 2 (military vehicles vs trucks), but mode 2 can do so.

In terms of constraints, we assume that there is a single resource pool of $R$ seconds to be used before all objects need to be classified. This number will also be varied across the experiments from 300 seconds to 700 seconds, to evaluate the bounds and algorithm performance for scenarios where the amount of sensor resources ranges from poor to rich.

In order to evaluate the utility of the lower bound, we compare the bound with the performance of two adaptive SM algorithms: a variation of Kastella's discrimination gain algorithm [5], which is a sequential algorithm for selecting the best sensor mode and target on the basis of maximizing the expected entropy reduction in the distribution of object type per unit sensor resource applied, and a dynamic SM scheduling algorithm based on Lagrangian relaxation and POMDP approximations described in [7]. The algorithms are summarized next.

The discrimination gain algorithm of [5] starts from the sufficient statistic $\underline{\pi}(k)$, consisting of the conditional probability type of each object after $k$ sensor actions have been taken. Associated with each object is the entropy of this distribution,

$$H(\pi_i(k)) = -\sum_{j=1}^{K} \pi_i(k) \log \pi_i(k) \tag{37}$$

For each sensor mode $m$ and each object $i$ such that the available resources allow the use of that mode, the expected entropy from using mode $m$ on object $i$ obtained from (11) as

$$E_y\{H(\pi_i(k+1)|y, i, m)\} = \sum_{y \in Y_m} P(y|i, m) H(\pi_i(k+1)|y, i, m) \tag{38}$$

The discrimination gain algorithm computes an index for each object and sensor mode, the expected entropy gain per unit resource, as

$$Gain(i, m)(k) = \frac{H(\pi_i(k)) - E_y\{H(\pi_i(k+1)|y, i, m)\}}{R_{im}} \tag{39}$$

and selects as its next sensing action the object and mode that has the highest $Gain(i, m)(k)$. Once all sensing resources are exhausted, the classification of each object is performed in a Bayes' optimal manner to minimize the expected classification cost.

The Lagrangian relaxation algorithm of [7] uses a receding horizon planning approach based on a POMDP algorithm that is similar to that used for computation of the lower bound, with the additional restriction that a maximum of three actions per object are considered. Since there is a resource constraint, the algorithm performs

a simple line search to vary a Lagrange multiplier. For each value of the Lagrange multiplier, a local SM policy is computed from (24) that uses a maximum of three sensing actions per object, using a variation of the Witness algorithm [12, 13]. The algorithm selects the best local SM policy found this way that satisfies the expected resource use bound in (20). This policy is likely to be conservative with respect to the use of resources, because to fully utilize the available resources requires the use of mixed local SM policies. A local SM policy can be viewed as a decision tree for each object, as in [7]. The initial action in such decision trees is deterministic, based on the current knowledge, and the future actions are contingent on the measured values. The Lagrangian relaxation algorithm computes the local SM policy, and schedules the initial sensing actions for each object. Once all the initial sensing actions are observed, the probability state is updated, the resource state is decremented, and the problem is solved again from the new probability state $\underline{\pi}$ and the remaining resource level. This process continues until no sensor resources remain to take additional sensing actions, at which time all of the objects are classified for minimum discrimination cost based on the available information.

Each algorithm was simulated for 100 independent Monte Carlo runs using the same measurement outcomes to evaluate its average performance for three different levels of sensor resources: 300 seconds, 500 seconds and 700 seconds. Figure 1 shows the results for the two algorithms and the lower bound for 300 seconds for a range of values of MD from 1 to 80. Figure 2 shows similar results for 500 seconds, and Figure 3 shows the results for 700 seconds. The results indicate that neither algorithm consistently performs close to the lower bound, but there are conditions where the performance of the algorithms and the lower bounds are close. For instance, when MD is close to 1, the costs of missed detections and false alarms is close, and policies such as maximizing information gain as measured by entropy are near-optimal. Similarly, the performance of the Lagrangian Relaxation algorithm is closer to the lower bound for limited sensor resources, as the limited lookahead approximation is closer to the actual optimal number of sensor actions per object. However, there is significant room for improvement in both policies: the discrimination gain algorithm fails to incorporate the relative values of different types of errors in its information seeking strategy, and the Lagrangian relaxation is conservative in that it does not use mixed strategies, and thus can underutilize sensor resources.

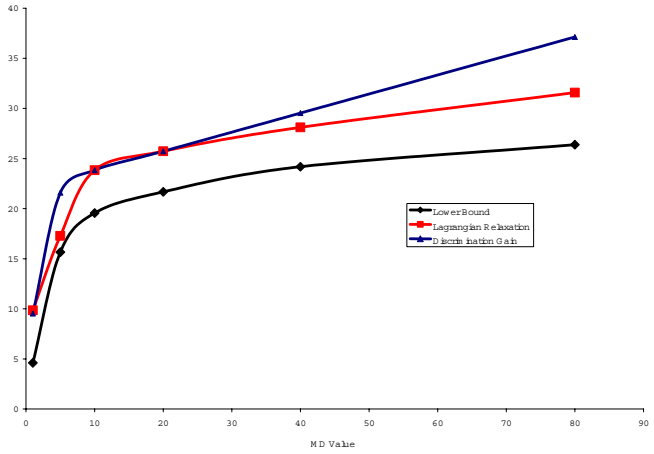


Figure 1: Monte Carlo performance of algorithms and lower bound for 300 seconds of sensor resource.

It is possible to construct curves similar to a receiver operating characteristic (ROC) by varying the value of $MD$. Such curves can characterize the potential tradeoffs in system performance achieved by different algorithms for a fixed amount of sensor resources. Figures 4, 5 and 6 illustrate the resulting ROC curves for 300, 500 and 700 seconds of sensor resources for the two algorithms. Note that the performance of the two algorithms is closer than the optimal values of Figs. 1-3 imply.

# 7 Discussion

In this paper, we have presented a mathematical formulation for adaptive multisensor management in problems of object classification as a partially observed Markovian decision problem. We developed an exact stochastic

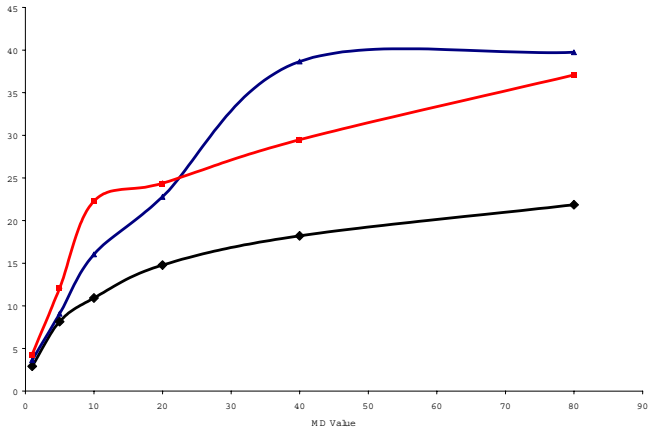


Figure 2: Monte Carlo performance of algorithms and lower bound for 500 seconds of sensor resource.

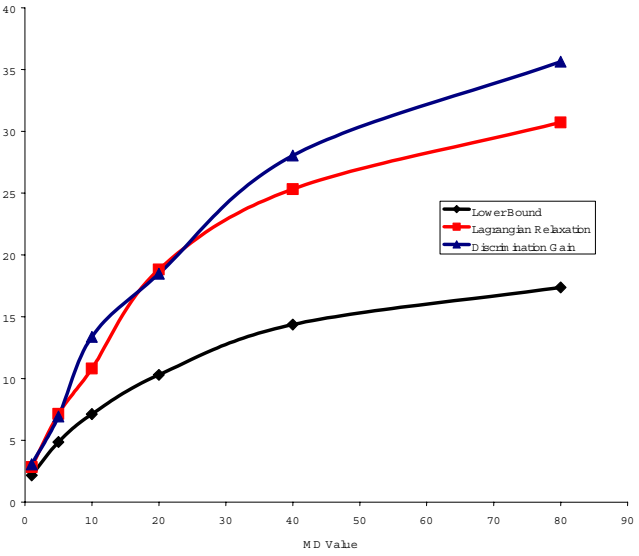


Figure 3: Monte Carlo performance of algorithms and lower bound for 700 seconds of sensor resource.

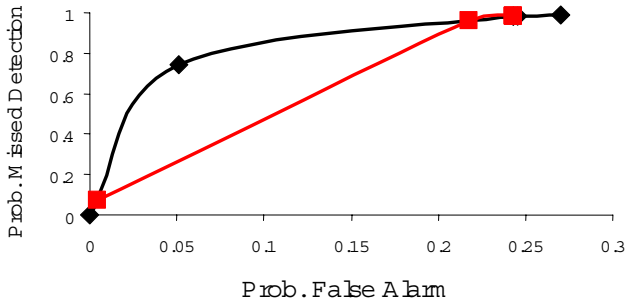


Figure 4: ROC of algorithms for 300 seconds of sensor resource.

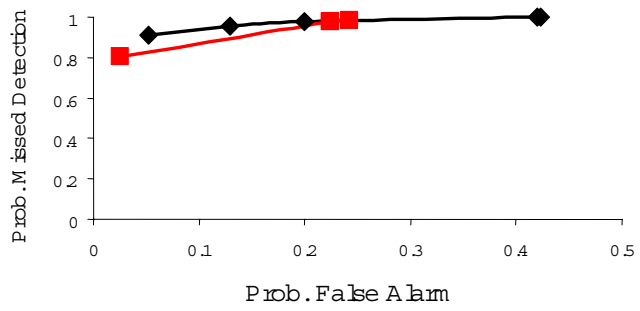


Figure 5: ROC of algorithms for 500 seconds of sensor resource.

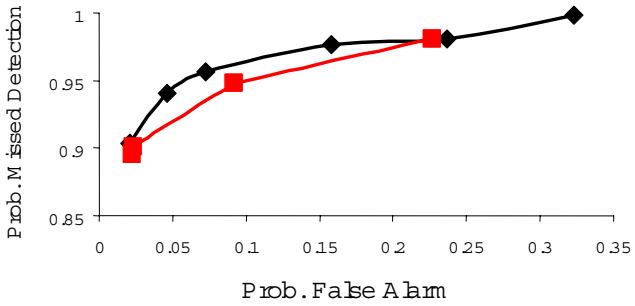


Figure 6: ROC of algorithms for 700 seconds of sensor resource.

dynamic programming algorithm for solution of these problems. However, the combinatorial nature of the decision space when multiple objects are present make the computations prohibitive even for small time horizons. We developed an approximate formulation that provides a lower bound on the achievable performance for such sensor management problems. This lower bound is obtained by expanding the space of admissible SM policies, replacing a sample path resource utilization constraint by an expected resource use constraint.

The resulting lower bound formulation is an integer programming problem, that has a simple, separable dual formulation. A key result in establishing this separability is to show that the lower bound formulation can be solved in terms of a subset of SM policies known as mixed local SM policies, which are random mixtures of policies that select actions on each object based only on the past information collected on that object. This results in a hierarchical algorithm for computing the lower bound, where dual variables are selected that decouple the SM problem into independent subproblems for each object. Each of the independent subproblems can be solved as a low-dimension partially observed Markov decision problem. The solutions of these independent subproblems are then used to improve the dual variables, until an optimal lower bound is obtained.

We presented experimental results that compared the lower bound with the performance of two suboptimal SM algorithms available in the literature. The experimental results established that the performance of both algorithms should be improved substantially in order to achieve the lower bound.

The lower bound developed in this paper can be used as a reference solution for the development of effective SM algorithms. Furthermore, the approximation used in developing the lower bound can be used in SM algorithms that attempt to optimize this lower bound, in order to generate practical real-time algorithms whose performance approaches this lower bound. This requires embedding the solution algorithms for lower bounds into a real-time algorithm such as model-predictive control, and developing a scheduling algorithm for determining when to recomputed the SM policies using a receding horizon approach. Development of real-time SM algorithms with performance that approaches the lower bound remains a challenge for future research.

# 8 Appendix

Outline of proof of Theorem 3.1, to be expanded later

Given an SM policy $\gamma$, construct a local behavior policy $\eta_i$ for each object $i$, which uses randomized decisions at each decision time, such that the marginal distribution of the decisions for object $i$ are the same under $\gamma$ and $\eta_i$, as follows:

Using policy $\gamma$, compute the marginal probability distribution of the first sensor action made on object $i$; note that this may include making no sensor action at all. Use this probability distribution as the probability distribution for selecting the first sensing decision in policy $\eta_i$. For each possible measurement value, compute the probability distribution of the next sensor action under policy $\gamma$ on object $i$. Use this probability distirbution as the distribution for selecting the second sensor action on object $i$, conditioned on the measurement obtained from the first action. Repeat this process until there are no sample paths generated by policy $\gamma$ with subsequent actions on object $i$. for the final classification decision on object $i$, compute the probability distribution of the final decision after the final sensing outcome on object $i$, aggregating over the sample paths generated by $\gamma$.

By construction, the marginal probability of sensor actions on object $i$ is the same under $\gamma$ and $\eta_i$. Repeating this construction for all objects $i$, one obtains a set of local SM random policies $\eta = \{\eta_1, \ldots, \eta_N\}$ that obtain the same expected classification performance and the same expected resource use as policy $\gamma$. From this random policy $\eta$, one can construct a mixed local SM policy with the same property, using the standard construction for generating mixed policies from random behavior policies in decision problems with perfect recall.

# References


[1] G. E. Monahan, "A survey of partially observable Markov decision processes: Theory, models and algorithms," *Mgmy. Sci.,* V. 28, p1-16, Jan. 1982.

[2] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable Markov processes over a finite horizon" *Op. Res.h,* V. 21, p 1071-1088, 1973.

[3] D. P. Bertsekas, *Dynamic Programming and Optimal Control,* Vols. I-II, Athena Scientific, Belmont, MA 1995.

[4] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1999.

[5] K. Kastella, "Discrimination Gain to Optimize Detection and Classification," *IEEE Trans. on Systems, Man and Cybernetics, Part A,* V. 27, No. 1, Jan. 1977.

[6] D. A. Castañón, "Optimal search strategies for dynamic hypothesis testing," *IEEE Trans. Sys., Man & Cybernetics,* v. 25, 1995.

[7] D. A. Castañón, "Approximate Dynamic Programming for Sensor Management," *Proc. 36th IEEE Conference on Decision and Control,* San Diego, CA, December 1997.

[8] D. A. Castañón and J. M Wohletz, "Model Predictive Control for Unreliable Dynamic Task Assignment," *Proc. 2002 Conf. Decision and Control,* Las Vegas, NV, Dec. 2002.

[9] G. Cohen, "Auxiliary problem principle and decomposition of optimization problems," *J. Opt. Theory and Appl.,* V. 32, 1980.

[10] W. S. Lovejoy, "A survey of algorithmic methods for partially observable Markov decision processes," *Annals of Operations Research,* v. 28, 1991.

[11] C. C. White, "Partially observed Markov decision processes: A Survey," *Ann. of Op. Res.,* V. 32, 1991

[12] M. L. Littman, A. R. Cassandra and L. Pack-Kaelbling, "Efficient dynamic programming updates in partially observable Markov decision processes," working paper, Brown University, Dec. 1995.

[13] A. R. Cassandra, *Exact and Approximate Algorithms for Markov Decision Processes,* Ph. D. Dissertation, Brown University, Providence, RI 1998.

[14] A. R. Cassandra, M. L. Littman and N. L. Zhang, "Incremental Pruning: A Simple, Fast Exact Method for Partially Observed Markov Decision Processes," *Proc. 13th Conf. Uncertainty in Aritifical Intelligence,* Providence, RI 1997.

[15] M. R. Garey and D.S. Johnson,*Computers and Intractability: A Guide to the Theory of NP-Completeness,* W.H. Freeman, New York, 1979.

[16] A. M. Geoffrion, "Lagrangian relaxation for integer programming," *Math. Prog. Studies,* v. 2, 1974.

[17] V. Krishnamurth and R. J. Evans, "Hidden Markov Model Multiarm Bandits: A Methodology for Beam Scheduling in Multitarget Tracking," *IEEE Trans. Signal Processing,* V. 49, N. 12, Dec. 2001.

[18] P. C. Gilmore and R. E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem" *Operations Research,* V. 9, 1961.

[19] V. M. Demyanov and L. V. Vasilev, *Nondifferentiable Optimization,* Optim. Software, New York 1985.

[20] S. J. Benkoski, M. G. Monticino, and J. R. Weisinger, "A Survey of the Search Theory Literature," *Naval Research Logistics,* Vol. 38, No. 4, 1991, pp. 469-494.

[21] K. A. Yost, *Solution of Large-Scale Allocation Problems with Partially Observed Outcomes,* Ph. D. Thesis, Naval Postgraduate School, Monterey, CA, Sept. 1998.

[22] K. A. Yost and A. R. Washburn, "The LP/POMDP Marriage: Optimization with Imperfect Information," *Naval Research Logistics,* Vol 47, No. 8, 607-619, 2000.

[23] K. A. Yost and A. R. Washburn, "Optimizing Assignments of Air-to-Ground Assets and BDA Sensors," *Military Operations Research,* Vol. 5, No. 2, 77-91, 2000.

[24] R. Chen and G. L. Blankenship, "Dynamic Programming Equations for Discounted Constrained Stochastic Control, " *IEEE Trans. Automatic Control,*" v.49, no. 5, May 2004.

# Closing the Loop in Sensor Fusion Systems: Stochastic Dynamic Programming Approaches

Michael K. Schneider, *Member, IEEE*, Gregory L. Mealy, *Member, IEEE*, and Felipe M. Pait, *Senior Member, IEEE*

***Abstract*—This paper provides an overview of the problem of managing sensor resources in a closed-loop sensor fusion system. We formulate the problem in a stochastic dynamic programming framework. In so doing, we expose structure in the problem resulting from target dynamics being independent and discuss how this can be exploited in solution strategies. We illustrate situations in which we believe such sensor management techniques are especially beneficial with two examples. One example is the management of a single sensor, and the other is the management of multiple sensors. The focus of both examples is on air-to-ground tracking.**

## I. Introduction

In this paper, we address control aspects of sensor fusion. For the sensor fusion problem of interest here, one would like to infer the state of multiple targets from measurements made by one or more sensors over time. Targets are typically located on the ground and can include vehicles, buildings, and other man-made objects. States of interest could include position, velocity, mode (e.g. on- or off-road), vehicle type, etc. Estimates of the states are inferred by fusing information from multiple sensors over time. The fusion engine responsible for piecing together information from different types of sensors will typically create hypotheses by associating new observations with previously detected targets. Alternative hypotheses are formulated to deal with ambiguities caused by incomplete or even contradictory information. New hypotheses are created and abandoned as data is accumulated that indicates the current target states have changed or resolves ambiguities in the past states of targets. The data can be generated by many different types of sensors, including airborne surveillance radars, video sensors, etc. The sensors are managed to collect the appropriate measurements. We view sensor resource management (SRM) as the control problem of allocating available sensor resources to obtain the best awareness of the situation.

This material is based upon work supported in part by the U.S. Air Force under Contract Nos. F33615-02-C-1197, F33615-03-M-1515, and F3365-02-C-1129.

The authors are with ALPHATECH, Inc., Burlington, MA 01803 USA. (phone: 781-273-3388; fax: 781-273-9345; e-mail: {michaels, gmealy, fpait}@ALPHATECH.com).

Efficient sensor management requires consideration of the value of particular pieces of information to the fusion engine at each moment, so the plant to be controlled comprises not only the sensors and communication systems, but also the fusion engine that processes the information collected by them, as illustrated in Fig. 1. The plant's inputs are precisely the requests that the sensor management system is allowed to make, and its outputs include all the information obtained from the sensors. The state of the plant is then the total information available to the fusion engine, and in principle also to the SRM controller, at a given time. The dimension of the state is not fixed: it increases as information is collected, and new tracks are initiated. It also decreases when new information results in hypotheses being resolved, and when the hypothesis tree is pruned of alternatives that are considered less likely.

From this point of view the process model is completely deterministic, and full information about the process is available. Uncertainty enters the picture in the form of the actual measurements obtained by the sensors, which can be treated as external disturbances about which we, as designers of a sensor management and fusion system, have no control or previous knowledge. Additional disturbances include sensor actions over which the system has no control – for example, sensor systems which are allocated at a higher command level. Indeed, the current state of the fusion system represents the best possible guess about the actual ground truth – taking into account the information available and our capacity to process it. Since the estimate does not depend on probabilities of obtaining specific data

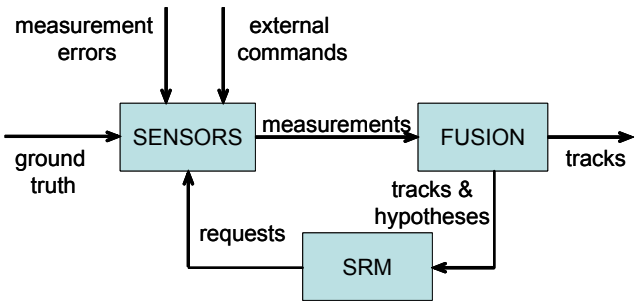


Fig. 1 Sensor Resource Management (SRM) closes the sensor/fusion control loop.

in the future, the system is essentially causal, a fact that simplifies conceptually the design of a sensor management algorithm. Of course the variable dimensionality of the state space precludes the use of textbook control design techniques, which are not likely to be applicable in any event.

A number of different approaches to the design of sensor managers have been proposed in the literature. They cover the different aspects of the sensor management problem including how to manage sensors to support detecting and localizing [3], [7], [8], [9]; tracking [2], [8], [10], [11], [12]; and classifying [4], [5], [6] targets. The proposed solutions include policies based on information-theoretic optimization criteria [8], [11] as well as policies for optimizing more traditional criteria (e.g., track error) generated using stochastic optimization techniques such as index rules [2], [5], [12]; Lagrangian relaxation [6]; et al. [3], [4], [7], [9], [10]. In this paper, we overview some of the technical issues in sensor management including structure in the problem that we believe can be exploited when designing solution techniques. This is discussed in a stochastic dynamic programming framework in Section II. In Section III, we illustrate situations in which we believe sophisticated sensor management strategies are especially beneficial with two examples. One example is the management of a single sensor, and the other is the management of multiple sensors. The focus of both examples is on air-to-ground tracking.

## II. Approximate Stochastic Dynamic Programming Approach

We have conceived designs to the sensor management control problem in the framework of stochastic dynamic programming. A typical formulation starts with the system state at time $t$, $x(t)$. The state includes all target true positions and types. A control at time $t$, $u(t)$, specifies a measurement of the system to be taken. The measurement may be corrupted by a stochastic disturbance $v(t)$ and may be delayed so that it is not realized until a later time. The measurement process is given by the function $h$, so that

$$y(t_y) = h(x(t), u(t), v(t)) \tag{1}$$

is the measurement realized at time $t_y > t$. The information about the system at time $t$ is summarized in the information state $I(t)$, consisting of all past measurements and controls

$$I(t) = \{y(t_y) : t_y \le t\} \cup \{u(t_u) : t_u < t\}. \tag{2}$$

The delay in realizing the measurement, $\Delta_y$, taken at time $t$, is a function of the information state, control, and stochastic disturbance at time $t$ so that

$$t_y = t + \Delta_y(I(t), u(t), v(t)). \tag{3}$$

Control decisions occur at discrete instants in time, $t_{u,0}$, $t_{u,1}$, $t_{u,2}$, .... Following time $t_{u,i}$, the next control is executed after the delay of $\Delta_u$, which is a function of the information state, control, and stochastic disturbance at time $t_{u,i}$. Thus,

$$t_{u,i+1} = t_{u,i} + \Delta_u(I(t_{u,i}), u(t_{u,i}), v(t_{u,i})). \tag{4}$$

The control is chosen from a constraint set $U(I(t))$ according to a control law, $\mu$, which is a function of the information state and time. Thus,

$$u(t_{u,i}) = \mu(I(t_{u,i}), t_{u,i}). \tag{5}$$

The sensor management policy is the collection of these control laws

$$\pi = \{\mu(I(t), t)\}. \tag{6}$$

Rewards are achieved upon executing the policy by attaining particular information states. The reward for attaining information state $I(t)$ is given by $R(I(t))$. These rewards are discounted by the factor $e^{-\gamma t}$ and integrated across time to yield an expected reward for executing policy $\pi$ from the information state $I(0)$ of

$$J_\pi(I(0)) = E\int_0^\infty e^{-\gamma\tau} R(I(\tau))\, d\tau. \tag{7}$$

The optimal sensor management policy $\pi^*$ is the one that maximizes (7) over all policies $\pi$. The optimal policy can be characterized in terms of Bellman's equation [1]. In this context, the equation states that the expected reward for the optimal policy satisfies

$$J^*(I(t)) = \max_{u(t)\in U(I(t))} \mathrm{E}\left[\begin{array}{l} \displaystyle\int_t^{t+\Delta_u(I(t),u(t),v(t))} e^{-\gamma\tau} R(I(\tau))\, d\tau + \\ J^*\left(I\left(t+\Delta_u(I(t),u(t),v(t))\right)\right) \end{array}\right]. \tag{8}$$

The first term on the right-hand side is the reward accrued until the next decision time after $t$. The second term is the expected reward after that time accrued from the resulting information state. The policy

$$\pi^* = \left\{\mu^*(I(t), t)\right\} \tag{9}$$

is optimal provided that the argument of the maximum in (8) is given by $\mu^*(I(t),t)$ for all $I(t)$ and $t$ (the assumption here is that the set of candidate controls is compact, if not finite, so that the maximum is well-defined). Several computational techniques, including both policy and value iteration, exploit the characterization in (8) to compute policies. The difficulties in exploiting this characterization are tied to the size of the state space, the set of candidate controls, and the set of stochastic disturbances. In particular, Bellman's equation characterizes $J^*$ for all possible information states $I(t)$ by evaluating the right-hand side of (8) for all possible controls $u(t)$, taking an expectation over all disturbances. This can be difficult to apply when the size of the sets involved is large.

However, there is special structure that can be exploited. Consider the following special case in which the system state is the aggregate state of $n$ targets

$$x(t) = \{x_1(t), \ldots, x_n(t)\} \quad (10)$$

whose individual states $x_i(t)$ are independent and evolving in time as Markov processes. This would be the case, for example, when tracking independent, isolated targets. Moreover, suppose the measurements of the system state are conditionally independent given target state and sensor controls so that one can write

$$y(t) = \{y_i(t) : i = 1 \ldots n\} \quad (11)$$

where an individual measurement can be written

$$y_i(t) = h_i(x_i(t), u(t), v_i(t)) \quad (12)$$

for independent stochastic disturbances $v_i(t)$. Independence introduces considerable structure; however, the problem is still complex since the information states of the system do not have similar independence properties. For example, one can consider partitioning the information state as

$$I(t) = I_1(t) \cup I_2(t) \cup \ldots \cup I_n(t) \cup I_u(t) \quad (13)$$

where

$$I_j(t) = \{y_j(t_y) : t_y \le t\} \quad (14)$$

and

$$I_u(t) = \{u(t_u) : t_u < t\}. \quad (15)$$

However, the future information states $I_j(\tau)$ for $\tau > t$ are neither independent nor conditionally independent given the current control $u(t)$ and system state $x(t)$. The reason is that the information states of targets are coupled through the control decisions. Thus, one cannot rely on methods for computing sensor management policies that require the independence of the targets' information states.

One approach we have used to develop sensor management policies that exploit the special structure is the application of index rules [1], [13]. Index rules are optimal for the following type of sensor management problem. There are $n$ targets, whose states are independent. A measurement can be made of only one target at a time, and the measurement is of fixed duration, i.e. $\Delta_y$ and $\Delta_u$ are constants and $\Delta_y < \Delta_u$. The state of the target can only change at instants when a measurement is made of it (e.g. the target state may not be changing, but the information state of the target may be as more measurements are acquired). In addition, the mission must be formulated such that the reward $R(I(t))$ accrued in a particular information state at time $t$ depends only on the information state $I_j(t)$ of the target $j$ being measured at that time. In this case, the optimal policy determining the next target at which to look from information state $I(t)$ is given by an index rule, which has the form

$$\mu(I(t)) = \arg\max_{j \in \{1,\ldots,n\}} m_j(I_j(t)) \quad (16)$$

where $m_j(I_j(t))$ is the index of the target. The index for target $j$ can be represented in terms of a single target problem. We have been able to develop solutions to these single target problems and apply the resulting index rule policy. Although the assumptions required for the index rule to be optimal are often violated in sensor management problems (e.g. one may be able to measure the state of more than one target at a time), we have found that index rules may still be optimal or, at least, applicable as part of heuristics [5], [14].

Another approach we have used to develop sensor management policies is to use limited lookahead algorithms [1]. A limited lookahead policy is one for which the control action is chosen as the solution to

$$\max_{u(t) \in U(I(t))} \mathrm{E}\left[\begin{array}{l} \int_{t}^{t+\Delta_u(I(t),u(t),v(t))} e^{-\gamma\tau} R(I(\tau)) d\tau + \\ \tilde{J}_1\left(I\left(t + \Delta_u(I(t), u(t), v(t))\right)\right) \end{array}\right]. \quad (17)$$

where

$$\tilde{J}_k(I(t)) = \max_{u(t) \in U(I(t))} \mathrm{E}\left[\begin{array}{l} \int_{t}^{t+\Delta_u(I(t),u(t),v(t))} e^{-\gamma\tau} R(I(\tau)) d\tau + \\ \tilde{J}_{k+1}\left(I\left(t + \Delta_u(I(t), u(t), v(t))\right)\right) \end{array}\right]. \quad (18)$$

for $k=1,\ldots,N-1$ where $N$ is the number of steps of lookahead and the terminal reward $\tilde{J}_N$ is chosen to approximate the expected reward. The algorithm for computing the limited lookahead policy is effectively enumerating possible controls and outcomes over $N$ steps, calculating a reward for the resultant state based on an approximation, and selecting the control that yields the best outcome. Structure in the problem can be exploited in the construction of $\tilde{J}_N$. As noted previously, the problem has special structure in that individual target state evolutions are often independent. One approach to exploiting this is to use an approximate terminal reward that is separable so that

$$\tilde{J}_N = \sum_{j=1}^{n} \tilde{J}_{N,j}(I_j(t) \cup I_u(t)) \quad (19)$$

for per-target rewards $\tilde{J}_{N,j}$. These can be constructed a number of different ways. One technique we have used is to calculate the expected rewards associated with a single-target form of the problem, motivated by the index definition in [13]. Essentially, we use a function of the index $m_j$ as the approximation $\tilde{J}_{N,j}$. We have also explored other methods for constructing $\tilde{J}_N$ including rollout and heuristic methods. In each case, we have tried to exploit structure in the problem such as the existence of independent target evolutions.

## III. Application Examples

What follows are two examples of how we have been applying these techniques to sensor management problems.

The examples outline how we have applied the stochastic control techniques described above to the development of sensor managers and illustrate areas where we have found distinct advantages to using these techniques. In order to illustrate the breadth of applicability, the examples are drawn from two different types of problems. The first is the control of a single sensor; the second is the control of multiple, distributed sensors.

### *A. Control of a Single Sensor*

In this first example, consider managing a single sensor air-to-ground radar tracking system. The sensor, tracker, and sensor manager are all colocated on the sensing platform. As a result, the latencies in transmitting information between components are minimal; so, the sensor manager is generating sensor controls on a fast time scale. In this context, two scenarios in which a stochastic control approach to sensor management has advantages are when there are differentiated targets and when the sensor mode must be matched to target state.

An example of the second scenario occurs when using an airborne radar to track ground targets. In the radar's standard ground moving target indicator (GMTI) mode, only targets moving against the background can be observed. However, the radar may have another mode such as a fixed-target indicator (FTI) mode, with which only stopped targets may be observed. In order to track the targets, the radar must be managed to periodically revisit targets in the appropriate mode to update the estimate of their position. Too long a period without observing the target will lead to the tracking system dropping the track. Longer track lifetimes are desirable. The sensor management problem is thus one of selecting the sequence of targets at which to look with the radar as well as the mode to use. One source of complexity in the problem is that targets may not be detected even if the appropriate mode is used. Thus, the sensor management policy must appropriately hedge to select the best mode based on past detections. Another potential source of complexity in the problem is that the measurements are taken over different durations $\Delta_y$ in the different modes. Thus, the policy must appropriately hedge in time so that longer duration modes are not chosen at poor instants in time. To address these two issues, we have developed a limited lookahead policy, of the form described by (17) and (18). The policy allows one to account for past detections as well as for predictions of future rewards that depend on the different measurement durations in each sensor mode. Initial results of performance are illustrated in Fig. 2. Here, a simple simulation is used to compute the average track lifetime for two different sensor policies. One is the limited lookahead policy; the other is a policy that only uses the GMTI mode. The simulation includes synthetic target motion, a simple tracker, and a simple sensor model. For this sensor model, the measurement durations are the same for the two different modes. The results indicate that constructing a sensor policy that takes advantage of the FTI sensor mode has the potential to provide significant improvements in track lifetime. More realistic simulations would be required to determine the precise benefit.

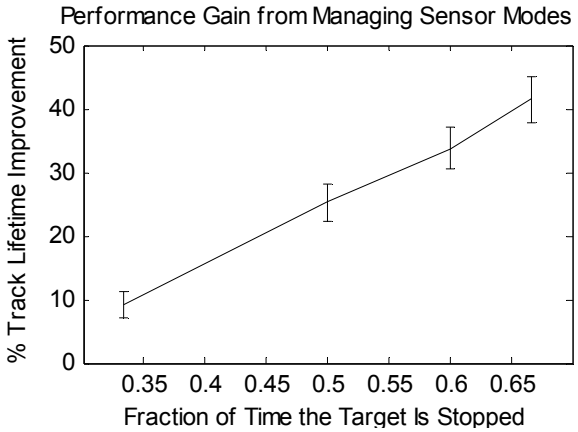


Fig. 2. The curve plots the fractional increase in track lifetime for using a limited lookahead sensor policy that controls sensor mode over a simple single mode policy. In this example, a different sensor mode must be used to observe the target when it is stopped than when it is moving. However, the sensor will only detect a target in the proper mode with some probability less than 1.

The other example of a scenario for which we have noted benefits of sensor management is one in which there are differentiated targets. Specifically, a subset of the tracked targets is designated by a user to be higher priority than the others. The high-priority targets could have different tracking requirements than the low-priority targets. For example, they may have more stringent track accuracy requirements. The specific context considered here is air-to-ground tracking with a GMTI radar. Thus, there is no mode selection problem for the sensor manager, as in the previously discussed scenario. However, the problem of selecting the sequence of targets at which to look is more complex. The sensor management policy must account for the different numbers of high-and low-priority targets, the different tracking requirements, and the current state of tracks to generate a control sequence that generates measurements of targets to meet the tracking requirements. Some initial, simple simulations indicate that significant benefits can be realized from a good sensor management policy. In particular, we simulated a scenario with a high-priority target and several low-priority targets. Two limited lookahead sensor management policies were evaluated. One used one step of lookahead (*N=1)*, and the other used two steps (*N=2)*. Both policies performed equally well on the high-priority target. However, the two step lookahead policy achieved track accuracy requirements on the low-priority targets 86% more of the time than the one step lookahead policy. This suggests that significant benefits can be realized by appropriately managing the sensor to

track differentiated targets. We are currently planning to evaluate the benefits in this type of scenario with a more realistic simulation.

### B. Control of Multiple, Distributed Sensors

The second example differs from the previous one in two key respects. The first is a decomposition of the sensor resource management function into two parts: an information valuation step followed by a sensor allocation calculation (i.e., constructing a sensor scheduling plan that maximizes the value of collected information subject to constraints on sensor availability and routing). The second is the introduction of multiple sensors into the problem. In this example, we focus on the information valuation aspect for multiple sensors of differing capability.

As described above, the fusion state is determined by both the stochastic evolution of the real system and the stochastic results of sensor measurements of that system. Different sensor tasking choices will thus result in different evolutions of the system's state. The decision becomes one of determining the optimal valuation of sensor resources with respect to their impact on the fusion process. While there are multiple reasons for requesting particular sensor tasks, the approach described herein addresses an important subset – requesting sensor tasks that will either improve target track estimates or remove association ambiguity in the current or near-future fusion state. To emphasize this aspect of the approach, the algorithm has been termed FIND (Fusion Information Needs Determination).

The goal of the FIND algorithm is to maximize the time discounted reward $J$ in (7) for the special case where the time between control actions $\Delta_u$ is constant so that one can rewrite it for a constant $\alpha$ as

$$J = \sum_{t=0}^{\infty} \alpha^t R(I(t)) . \tag{20}$$

The reward function $R$ has the form

$$R(I(t)) = \sum_j R_j(I_j(t) \cup I_u(t)) \tag{21}$$

where the index $j$ in this case ranges over the hypothesis space of the fusion system. The track hypothesis space contains information about the relative certainty of different data associations that are not reflected in the single global set of track estimates normally output. The individual rewards $R_j$ are a function of a set of goals and priorities, specifically:

1. The required kinematic accuracy (expressed as tracking uncertainty, $\Sigma_{Goal}$) for tracking confirmed targets
2. The required classification accuracy (probability of correct classification, $P_{Goal}$) for declaring high confidence identification of a target
3. The relative priorities for meeting the kinematic and classification accuracy goals, both singly and in combination, for each of the expected target types
4. Indications of time-criticality of the information need.

Given this information, we can specify the reward for a given hypothesis $H_j$. The reward takes on differing values depending upon which combination of the goals is satisfied. For hypothesis $H_j$, with associated kinematic uncertainty $\sigma_j^2(t)$ (the maximum eigenvalue of the position error covariance) and classification probabilities $p_j(t)$ (defined as the vector of probabilities that the target is of a given type), the individual reward at time t is given by:

$$R_j(I_j(t) \cup I_u(t)) = \begin{cases} 0 & \sigma_j^2(t) > \Sigma_{Goal} \text{ and } \max(p_j(t)) < P_{Goal} \\ R_\Sigma & \sigma_j^2(t) \le \Sigma_{Goal} \text{ and } \max(p_j(t)) < P_{Goal} \\ R_Y & \sigma_j^2(t) > \Sigma_{Goal} \text{ and } \max(p_j(t)) \ge P_{Goal} \\ R_{Y\Sigma} & \sigma_j^2(t) \le \Sigma_{Goal} \text{ and } \max(p_j(t)) \ge P_{Goal} \end{cases}$$

Different candidate sensor tasks are valued using a 1-step limited lookahead approach given by (17) and (18). A heuristic approximation of the terminal award is given by the separable function

$$\tilde{J}_1 = \sum_j (p_j^T \rho\, p_j + R_\Sigma) \left( \frac{1}{\pi} \arctan \frac{(\Sigma_{Goal} - \sigma_j^2)}{\Gamma} + 0.5 \right) \tag{22}$$

where the summation is over the different hypotheses within the track hypothesis space. The FIND values are computed as the increment in the expected reward of the one-step lookahead for a set of candidate sensor tasks

$$\underbrace{V(I(t), u(t), \Delta_u)}_{\text{FIND value}} = \underbrace{\mathrm{E}\left[ \tilde{J}(I(t) \cup y(t+\Delta_u) \cup u(t)) \right] - \tilde{J}(I(t))}_{\text{Incremental reward}} \tag{23}$$

Since FIND does not have information as to which specific sensors are available, the FIND value is computed for a set of candidate sensor controls $u$ parameterized by hypothesis as well as a range of kinematic measurement accuracies and classification abilities.

The FIND valuation is used to determine the benefit derived from tasking a sensor to provide information on a specified hypothesis. In practice, these valuations are rank-ordered and filtered such that only a subset of the possible hypotheses is considered in the sensor allocation calculation. This portion of the solution balances the set of valuations (which vary with sensor performance) against competing requirements (e.g., requests produced at a higher command level) to produce a multiple sensor tasking plan.

To illustrate the performance of the FIND algorithm and demonstrate its utility for identifying (and quantifying) the benefits of candidate sensor taskings, consider the simple scenario. It begins with a single, stationary, high priority target. Initial information about the target consists of good classification, but poor kinematic information. A short time later, two distinct tracks are reported by an MTI system.

While these reports provide good kinematic information, target classification knowledge is poor. The problem becomes one of identifying which, if either, of the moving targets is the original high priority one.

Three approaches for generating sensor task valuations were examined:

1. **Raster** which simply tasks the sensor(s) to address each hypothesis in turn
2. **Myopic** which implements a 1-step lookahead, greedy approach. Defaults to Raster if no sensor task is expected to achieve a goal
3. **FIND** which implements a 1-step lookahead and uses the heuristic terminal reward in (22) to approximate the long-time reward.

Two sensors are available for tasking. Nominally, the first provides accurate kinematic information but no classification data, while the second provides classification information but has a poorer (i.e., larger) kinematic uncertainty. Each is assumed to report the results of the tasked observation. The FIND problem is to produce a sensor tasking (or set of sensor tasks) that resolves the inherent ambiguities in the hypothesis space while minimizing the number of such tasks. This is equivalent to producing a set of recommended sensor taskings that results in the best (minimum number of observations) solution to achieving the target tracking and classification goals.

Fig. 3 illustrates how the above algorithms perform for one set of evaluation conditions. The curves are the probability that the tracking and classification goals are exceeded as a function of the number of recommended sensor taskings. The results shown in the figure are representative; the FIND algorithm is clearly superior to the other approaches.

The valuations provided by the FIND algorithm can be viewed as providing different types of requests to improve fusion performance. The highest value requests are those which remove ambiguity in report associations to high priority targets. Requests that confirm ID and track likely high priority targets typically have medium values, while those with the lowest value are requests to ID unknown targets and are usually ignored unless no higher value tasks are requested for a given sensor resource.

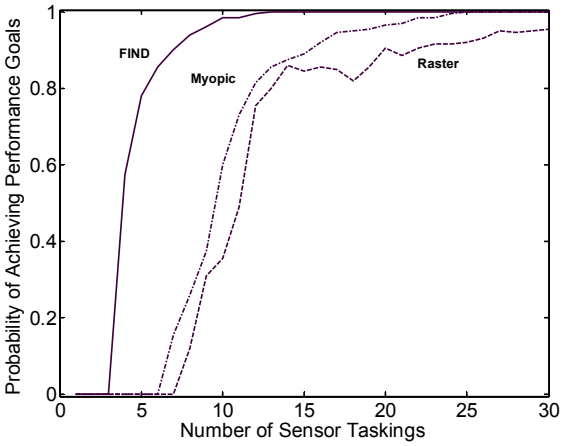


Fig. 3. FIND significantly reduces the number of sensor taskings required to achieve performance goals.

## IV. Conclusion

The examples in the previous section highlight issues in sensor management and indicate how one could exploit structure resulting from independent target motions to develop a sensor management policy. The results indicate that such policies will appropriately allocate sensor resources to improve the resolution of hypotheses in multi-target tracking systems and, specifically, to improve the surveillance of high-priority targets. Further experimentation is required to determine the precise degree to which benefits can be realized in practice. Planned development of high fidelity simulations will allow us to perform the necessary experiments. We expect results will indeed confirm that significant benefits can be realized.

## References


[1] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 2001.

[2] V. Krishnamurthy and R. J. Evans. Hidden Markov model multiarm bandits: A methodology for beam scheduling in multitarget tracking. *IEEE Transactions on Signal Processing*, 49(12):2893 - 2908, 2001.

[3] S. Singh and V. Krishnamurthy. The optimal search for a Markovian target when the search path is constrained: The infinite-horizon case. *IEEE Transactions on Automatic Control*, 48:493-497, 2003.

[4] V. Krishnamurthy. Algorithms for optimal scheduling and management of hidden Markov model sensors. *IEEE Transactions on Signal Processing*, pages 1382 - 1397, 2002.

[5] D. A. Castanon. Optimal search strategies in dynamic hypothesis testing. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(7):1130-1138, 1995.

[6] D.A. Castanon. Approximate dynamic programming for sensor management. In *Proceedings of the 36th IEEE Conference on Decision and Control*, 1997.

[7] B. LaScala, B. Moran, and R. Evans. Optimal adaptive waveform selection for target detection. In *Proceedings of the International Radar Conference*, 2003.

[8] E. Ertin, J. W. Fisher, and L. C. Potter. Maximum mutual information principal for dynamic sensor query problems. In *Proceedings Information Processing in Sensor Networks*, pages 405-416, 2003.

[9] D. Sinno, D. Cochran, and D. Morrell. Multi-mode detection with markov target motion. In *Proceedings of the 3rd International Conference on Information Fusion*, pages 25-31, 2000.

[10] D. Sinno and D. Cochran. Dynamic estimation with selectable linear measurements. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2193-2196, 1998.

[11] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration. *IEEE Signal Processing Magazine*, 19:61-72, 2002.

[12] R.B. Washburn, M.K. Schneider, and J.J. Fox. Stochastic dynamic programming based approaches to sensor resource management. In P*roceedings of 5th International Conference on Information Fusion*, pages 608- 615, 2002.

[13] J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):148-177, 1979.

[14] P. Whittle. Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability*, 25A:287--298, 1988.

# Farsighted Sensor Management Strategies for Move/Stop Tracking

**Angelia Nedich, Michael K. Schneider, and Robert B. Washburn**

BAE Systems
6 New England Executive Park
Burlington, MA, USA.
{angelia.nedich, michael.k.schneider}@baesystems.com

***Abstract –*** *We consider the sensor management problem arising in using a multi-mode sensor to track moving and stopped targets. The sensor management problem is to determine what measurements to take in time so as to optimize the utility of the collected data. Finding the best sequence of measurements is a hard combinatorial problem due to many factors, including the large number of possible sensor actions and the complexity of the dynamics. The complexity of the dynamics is due in part to the sensor dwell-time depending on the sensor mode, targets randomly starting and stopping, and the uncertainty in the sensor detection process. For such a sensor management problem, we propose a novel, computationally efficient, farsighted algorithm based on an approximate dynamic programming methodology. The algorithm's complexity is linear in the number of targets. We evaluate this algorithm against a myopic algorithm optimizing an information-theoretic scoring criterion. Our simulation results indicate that the farsighted algorithm performs better with respect to the average time the track error is below a specified goal value.*

**Keywords:** Tracking, sensor management, farsighted strategy, stochastic dynamic programming.

## 1 Introduction

We consider the problem of sensor management arising in tracking multiple targets with a multi-mode sensor. The sensor management problem is to determine which target to be observed by the sensor and which sensor mode to use. These decisions are to be made over time so as to optimize the utility of the collected measurements. An example of the sensor management problem of interest is that of managing a multi-mode airborne radar to track moving and stopped ground targets. Specifically, the radar may have two modes, a moving target indicator (MTI) mode for observing moving targets and a fixed-target indicator (FTI) mode for observing stopped targets. Each mode is characterized by the uncertainties in the target detection and measurement processes as well as the measurement collection time. These may all be different in the different modes. The radar controls include which mode to use as well as where to point the radar for a particular measurement. The objective is to collect enough data on target position over time to meet desired track error goals. Determining how to optimally manage a sensor over time to meet such objectives is a hard combinatorial optimization problem. The problem complexity stems from many factors, including the large number of possible sensor actions as well as the complexity of the dynamics. The complexity of the dynamics results partially from the radar dwell-time depending on the radar mode, targets randomly starting and stopping, and the uncertainty in the sensor detection process. As a result of these factors, computing optimal sensor management strategies is often infeasible.

Various strategies have been proposed for sensor management including strategies that use information-theoretic scoring criterions such as those developed in [1] for tracking, and [2] and [3] for collaboration of networked sensors. Some farsighted strategies have been developed in [4], [5], and [6] for tracking. More specifically, the work in [6] evaluates some farsighted strategies and compares them to a myopic strategy. A farsighted strategy is one where the sensor manager considers the benefits resulting from a sequence of (two or more) sensor actions, while a myopic strategy is one where the sensor manager considers only the benefits resulting from a single sensor action. In [6], the evaluations of these two kinds of strategies are performed on the problem of managing a single mode sensor to track targets that may become occluded.

The work presented in this paper is also motivated by an interest in comparing the benefits of farsighted strategies with that of myopic strategies. In contrast to previous work, the investigation presented here considers a novel farsighted algorithm and problem for evaluation. In particular, we consider the problem of move/stop tracking outlined above, which has not previously been

considered for evaluating the relative benefits of farsighted sensor management strategies. For this problem, we develop a novel, efficiently computable farsighted sensor management strategy.

In the remainder of this paper, we present the algorithms for evaluation and the results of that evaluation. In Section 2, we describe the farsighted sensor management strategy, which is based on an approximate stochastic dynamic programming methodology. In Section 3, we introduce a myopic sensor management algorithm that serves as a baseline for the evaluation of the farsighted sensor management algorithm. In Section 4, we report some simulation results comparing the two algorithms. In Section 5, we give some concluding remarks.

# 2 Farsighted Strategy

Our development of the farsighted strategy is based on a formulation of the sensor management problem of interest as a stochastic dynamic program. This formulation is discussed in the following section.

## 2.1 Formulation As a Dynamic Program

We formulate our sensor management problem as a continuous-time stochastic dynamic programming problem with an infinite horizon (cf. [7]). The system to be controlled is the tracker, whose state $x(t)$ at time $t$ is composed of the target track states $x_i(t)$, i.e.,

$$x(t) = \left( x_1(t), \ldots, x_n(t) \right),$$

where $n$ is the number of targets. The target track state $x_i(t)$ is given by

$$x_i(t) = \left( S_i(t), p_{im}(t), p_{is}(t) \right),$$

where $S_i(t)$ is the position-error covariance for target $i$, and $p_{im}(t)$ and $p_{is}(t)$ are the probabilities that the target is moving or is stopped at time $t$, respectively. These probabilities can be estimated, for example, by computing the target-mode likelihoods based on the target detection history.

The candidate controls include the location of a sensor dwell and the mode of the sensor. For clarity of exposition, we assume that *the sensor may observe only one target at a time*. In particular, we say that the sensor observes target $i$ when the sensor is actually pointed at the estimated position of target $i$. Therefore, at any state and time, the set of candidate controls $U$ consists of target-radar mode pairs, i.e.,

$$U = \left\{ (i, m) \mid i \in \{1, \ldots, n\}, m \in \{MTI, FTI\} \right\}.$$

This set can be modified to include other candidate locations to accommodate a more general case when the sensor may observe more than one target at a time.

Given that a control $u(t_k)$ has been selected at time $t_k$, the new state of the system depends on the control $u(t_k)$ and the resulting measurement outcome. The dynamics of the state are given by a track filter, such as an interacting multiple-model (IMM) filter (see [8]). To simplify the notation, for the target-state and system-state evolution, respectively, we will write

$$x_i(t) = f\left( x_i(t_k), u(t_k), w, t - t_k \right),$$

$$x(t) = F(x(t_k), u(t_k), w, t - t_k),$$

where $w$ is the detection outcome with $w$=1 or $w$=0 indicating detection and missed detection, respectively.

Now, we define a reward $r_i$ for each target $i$, which is nonzero only when the target-track error is sufficiently small. In particular, for a specified track error goal $G_i$ associated with target $i$, the reward $r_i$ is nonzero if the mean-squared track error, the trace of the error covariance, is below $G_i$. Otherwise, $r_i$ is zero. Formally,

$$r_i(x_i(t)) = \begin{cases} V_i & \text{if } Tr(S_i(t)) \le G_i, \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where $V_i$ is the priority value for target $i$ and $Tr(\cdot)$ denotes the trace. In essence, the reward is collected only when the track accuracy is high enough. The total reward $R$ at state $x(t)$ is the sum of the target rewards,

$$R(x(t)) = \sum_{i=1}^{n} r_i(x_i(t)).$$

The sensor manager goal is to select the sequence of controls $\{u^*(t_k) | k=0,1,\ldots\}$ maximizing the expected reward accumulated over time. Formally, the sensor manager is solving the following problem

$$\max_{\{u(t_k) | k=0,1\ldots\}} E\left[ \int_0^{\infty} R(x(t)) e^{-gt} dt \mid x(0) = x_0 \right],$$

where ? is a decay factor modeling the information depreciation in time and $x_0$ is the initial state of the system.

According to dynamic programming theory, there is an optimal policy $p^*$ specifying an optimal control for each state. Furthermore, every optimal policy satisfies Bellman's equation:

$$J^*(x) = \max_{u \in U} \left\{ \int_0^{t_u} R(x(t)) e^{-gt} dt + e^{-gt_u} \underset{w}{E} J^*(F(x, u, w, t_u)) \right\},$$

where $t_u$ is the time needed to complete the sensor task specified by a control $u$, and $J^*(x)$ is the expected reward accumulated under the optimal policy $p^*$ when the system starts at state $x$. Note that, to find an optimal policy using Bellman's equation, the equation has to be solved for all states $x$. When the number of states is large or the number of control choices is large, solving Bellman's equation is computationally prohibitive due to the combinatorial explosion of state-control combinations. This is exactly the case with our sensor management problem, where the computational complexity grows exponentially with the number of targets, the number of target track states, the number of sensor modes, and the number of sensor dwells. For example, the number of target track states grows exponentially with the number of sensor dwells, and can be large even for a single target. Many different techniques for generating suboptimal policies have been developed. We consider a rollout-based approach, as discussed in the following section.

## 2.2 Rollout Strategy

The rollout strategy is an approximate dynamic programming technique (see [7], Vol. 1, p. 314). This technique evaluates a control action by estimating near and far-future benefits resulting from the control choice at the current state. The near-future benefits are computed by predicting the action consequences over the look-ahead planning stages. The far-future benefits are the benefits accumulated after the look-ahead stages. In the rollout approach, the far-future benefits are computed as the benefits resulting from applying a fixed policy. This approach as illustrated in Figure 1.

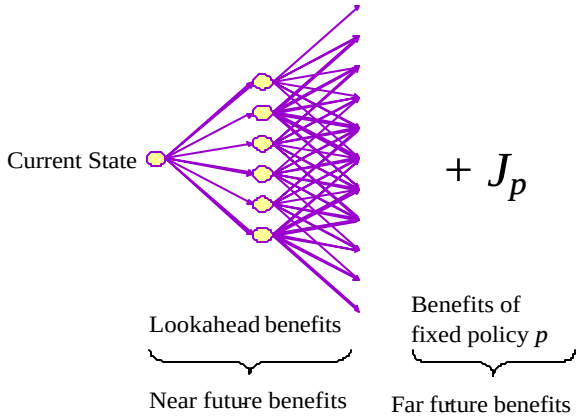


Figure 1. The rollout approach evaluates near and far-future benefits of an action taken at the current state.

Formally speaking, the rollout approach is one policy-iteration step performed on some fixed policy $p$. In particular, for a fixed policy $p$, one step of the policy-iteration consists of solving the following problem:

$$\max_{u \in U} \left\{ \int_0^{t_u} R(x(t)) e^{-gt} dt + e^{-gt_u} \underset{w}{E} J_p(F(x, u, w, t_u)) \right\} \quad (2)$$

where $J_p$ is the expected reward of policy $p$. The resulting overall policy is a one-step improvement of the original policy $p$. Thus, it is desirable that $p$ is near-optimal. Furthermore, it also desirable to select $p$ so that the expected policy-reward $J_p$ is computable.

In the rollout-based algorithm, the sensor manager selects a sensor action by solving (2). As a first step toward solving this maximization problem, we discretize the time by letting

$$t_k = kd \quad \text{for } d > 0 \text{ and } k = 0, 1, 2, \ldots .$$

Then, relation (2) reduces to

$$\max_{u \in U} \left\{ \sum_{k=0}^{K_u - 1} \alpha^k R(x(t_k)) + \alpha^{K_u} \underset{w}{E} J_p(F(x, u, w, t_{K_u})) \right\} \quad (3)$$

where $K_u$ is the sensor dwell-time (in units of $d$) for the sensor action specified by control $u$, and $\alpha \in (0,1)$ is a discount factor given by

$$\alpha = e^{-\lambda d}.$$

Recall that for any control choice $u=(j,\mu)$, the detection outcome $w$ can take value 1 or 0. Thus, the expectation in relation (3) reduces to

$$\begin{aligned} \underset{w}{E}\, J_p(F(x,u,w,t_{K_u})) = \\ p_{jm}(t_{K_u})\, b_{jm}\, J_p(F(x,u,1,t_{K_u})) \\ +\left[1-p_{jm}(t_{K_u})\, b_{jm}\right] J_p(F(x,u,0,t_{K_u})), \end{aligned}$$

where $p_{jm}(t_{K_u})$ is the probability that target $j$ is in mode $\mu$ at the time of the observation, and $ß_{j\mu}$ is the detection probability for radar mode $\mu$ when observing target $j$.

To accommodate efficient computation of the expected policy-reward, our sensor resource manager uses a tracker predictive model approximating the tracker. This model is based on the following:

*Assumption*:

1. Each target is either moving or is stopped, but the target motion state is unknown.

2. A target track is dropped if the target is not detected.

Assumption 1 is realistic for cases where the changes in target motion take longer than planning and executing a sensor action. Assumption 2 is more conservative than necessary (a target track may continue even with one or more missed detections). However, the resulting model is useful for planning purposes. Furthermore, these assumptions restrict the branching of the control-outcome space of any policy. This allows us to evaluate our farsighted policy without using costly Monte Carlo simulations.

We now discuss the choice of policy $p$. Motivated by the desire to have a good policy whose expected reward can be computed for any initial state, *we consider a policy p having the following properties*:

1. A target is observed with either MTI or FTI mode at all times.

2. Initially, the targets are sorted in a list according to some criterion. Then, these targets are observed according to the list as follows: each target is observed until either its track error decreases below the desired value or its track is dropped. If the track error is decreased below the desired value, the target is revisited at the rate that keeps its track error below the desired value.

We assume that the sensor can revisit the targets with rates that keep the track errors below the desired values.

We next describe a procedure for selecting a radar mode and a procedure for ordering the targets in a list. In particular, let $x$ be the system state when the policy $p$ is to be applied, i.e.,

$$x=(x_1,\ldots,x_n) \quad \text{with} \quad x_i=(S_i,\, p_{im},\, p_{is}),\, i=1,\ldots,n.$$

From the current target-mode probabilities $p_{im}$ and $p_{is}$, we determine the most likely target modes

$$m(i)=\operatorname{argmax}\{\, p_{im},\, p_{is}\,\}.$$

Under the policy $p$, the radar always uses mode $\mu(i)$ when observing target $i$. Given the current target covariances $S_i$, $i$=1,…,$n$, the policy $p$ sorts the targets according to the vicinity of their track errors to the desired goal, i.e., the targets are sorted according to the values $Tr(S_i)/G_i$ for $i$=1,...,$n$. This order is motivated by that generated according to an index-rule policy such as that discussed in [5].

We now evaluate the policy reward $J_p$. We note that, even for our simple tracker predictive model, it is still computationally prohibitive to exactly evaluate the reward $J_p$ due to coupling of the target dynamics under the policy. We thus approximate the policy reward $J_p$ assuming that it is separable across the targets, i.e.,

$$J_p(S)=\sum_{i=1}^{n} J_i(S) \qquad \text{with} \quad S=(S_1,\ldots,S_n).$$

For notational convenience, without loss of generality, we may assume that the order of the targets is 1, 2,…, $n$ when they are sorted according to values $Tr(S_i)/G_i$. Suppose that the mean-squared errors for the first *?* targets are within their goal region, i.e.,

$$Tr(S_i)\le G_i \qquad \text{for } i=1,\ldots,k.$$

For these targets, the reward $J(S_i)$ is the reward collected during the revisit period, i.e.,

$$J_i(S)=L_i \qquad \text{for } i=1,\ldots,k,$$

where $L_i$ is the long-term reward accumulated during target revisits (to be discussed shortly).

Consider now the targets $i$=*?*+1,..,$n$. Their mean-squared errors exceed the desired goals. Under the policy $p$ (property 2), each of these targets is observed until the

track is dropped or its mean-squared error decreases below the value $G_i$. However, it can be seen that the accumulated reward is nonzero only if the mean-squared error decreases below the value $G_i$. Let $T_{?+1}$ be the observation time required to decrease the trace $Tr(S_{?+1})$ below value $G_{?+1}$. Then, we have

$$J_{k+1}(S) = \left(ab_{k+1,m(k+1)}\right)^{T_{k+1}} L_{k+1}.$$

While observing target ?+1, the covariances of targets ?+2,…,$n$ evolve to $S_{?+2}(T_{?+1}),\ldots,S_n(T_{?+1})$. Let $T_{?+2}$ be the observation time required to decrease the trace of covariance $S_{?+2}(T_{?+1})$ below value $G_{?+2}$. Then, we have

$$J_{k+2}(S) = \left(ab_{k+2,m(k+2)}\right)^{T_{k+1}+T_{k+2}} L_{k+2}.$$

Continuing in this manner, we can see that

$$J_i(S) = \left(ab_{i,m(i)}\right)^{T_{k+1}+T_{k+2}+\ldots+T_i} L_i \quad \text{for } i = k+1,\ldots,n.$$

where $T_i$ is the time required to decrease the trace of $S_i(T_{?+1}+\ldots+T_{i-1})$ below goal value $G_i$.

We next discuss the long-term rewards $L_i$ accumulated during periodic revisits of the targets. As mentioned earlier, once the traces of error covariances for all targets decrease below their corresponding goal values, the targets are revisited at a constant rate. Under Assumption 1, the target modes do not change in time, so the error covariance of a stopped target does not change in time. Therefore, the stopped targets with error traces below their corresponding goals are not revisited, and the long-term reward $L_i$ is given by

$$L_i = \sum_{t=0}^{\infty} a^t V_i = \frac{V_i}{1-a} \quad \text{for a stopped target } i.$$

We focus now on the long-term reward $L_i$ associated with a moving target $i$. Let $M$ be the length of the revisit interval required for keeping the trace of the target $i$ error covariance below the desired value. Without loss of generality we may assume that the sensor revisits the target $i$ at times $t=jM$, $j$=0,1,…. During the revisit process, the reward is collected for as long as the target is detected, and the reward ceases when the target is not detected (cf. Assumption 2). Thus, during the revisit stage, the reward is collected per unit of target lifetime. The target lifetime is a random variable taking value $jM$ with probability $b_{i,m(i)}^{j-1}\left(1-b_{i,m(i)}\right)$. Hence, if the lifetime takes value $jM$, then the collected reward $G(jM)$ is equal to

$$\Gamma(jM) = r\left(1+a^M+\ldots+a^{(j-1)M}\right) = r\frac{1-a^{jM}}{1-a^M},$$

where ? is the reward accumulated between any two consecutive revisits and is given by

$$r = V_i\left(1+a+\ldots+a^{M-1}\right) = V_i\frac{1-a^M}{1-a}.$$

The long-term reward $L_i$ is equal to the expected reward G collected during the target lifetime, so that by computing the expected value of $G(jM)$, we can see that

$$L_i = \frac{V_i\left(1-a^M\right)}{\left(1-a\right)\left(1-a^M b_{i,m(i)}\right)} \quad \text{for a moving target } i.$$

Concluding this section, we note that the preceding algorithm description involved only two sensor modes mainly for the clarity of exposure. The algorithm can be easily extended to a more general case involving an arbitrary number of sensor modes. Note that, the preceding procedure for approximating the expected reward $J_p$ has polynomial complexity in the number of targets, in contrast to the exponential complexity required to compute the reward $J_p$ exactly.

# 3 Myopic Strategy

Here, we present a myopic sensor management algorithm that serves as a baseline for evaluating the performance of the farsighted algorithm discussed in the preceding section. We do not consider a myopic approach optimizing the dynamic programming formulation presented in the previous section. This is because the reward function has a threshold structure so that no value may be realized from a single action. Instead, we consider an algorithm that evaluates sensor actions based on the expected decrease in the entropy of the target-track errors per unit of time. The algorithm is myopic since the changes in the entropy are computed only for a single sensor action.

Specifically, the entropy $h_i$ for target $i$ at state $x_i=(S_i,p_{im},p_{is})$ is given by

$$h_i(x_i) = \frac{V_i}{2}\log\left[2p_c eTr(S_i)\right], \qquad (4)$$

where $V_i$ is the priority of target $i$ and $p_c$~3.14 (see [9], Chapter 9). As seen from this relation, the target entropy is measured by the target-track error in log-scale. The entropy $H$ of the system at state $x$ is defined as the sum of the target entropies $h_i$:

$$H(x) = \sum_{i=1}^{n} h_i(x_i) \quad \text{where } x = (x_1, \ldots, x_n). \tag{5}$$

Let $t_k$ be the current time and $x(t_k)$ be the current state. Given a control $u$ is selected at time $t_k$, the expected entropy decrease per unit of time at state $x(t_k)$ is given by

$$D(x(t_k), u) = \frac{E_w\left[H(x(t_{k+1}))\right] - H(x(t_k))}{t_u},$$

where $t_u$ is the time required for completing the sensor action specified by control $u$, $t_{k+1} = t_k + t_u$, is the time the detection outcome $w$ is available, and $x(t_{k+1})$ is the state to which the system transitions under the control $u$. From relations (4) and (5), it can be seen that for state $x(t_k)$ with components

$$x_i(t_k) = \left(S_i(t_k), p_{im}(t_k), p_{is}(t_k)\right)$$

and for control $u=(j,\mu)$, the expected entropy decrease is

$$\begin{aligned} D(x(t_k), u) = {} & \frac{1}{2t_u} \sum_{i=1}^{n} V_i \log \frac{Tr\left(S_i^+(t_{k+1})\right)}{Tr\left(S_i(t_k)\right)} \\ & + \frac{1}{2t_u} b_{jm} p_{jm}(t_k) V_j \log \frac{Tr\left(\hat{S}_j(t_{k+1})\right)}{Tr\left(S_j^+(t_{k+1})\right)}, \end{aligned} \tag{6}$$

where $S_i^+(t_{k+1})$ is the predicted target covariance and $\hat{S}_i(t_{k+1})$ is the updated target covariance according to the standard Kalman-filter equations.

The entropy-based sensor manager is myopic: at state $x(t_k)$, it selects a control $u^*$ optimizing the entropy decrease $D$. More specifically, the entropy-based sensor manager solves the following problem

$$\min_{u \in U} D(x(t_k), u),$$

where $D(x(t_k),u)$ is computed according to equation (6).

# 4 Numerical Results

Here, we present our simulation scenario and the test results obtained for the farsighted rollout algorithm and the myopic entropy-optimizing algorithm. The purpose of the simulations is to determine if the farsighted algorithm has any potential advantages over the myopic algorithm.

## 4.1 Simulation scenario

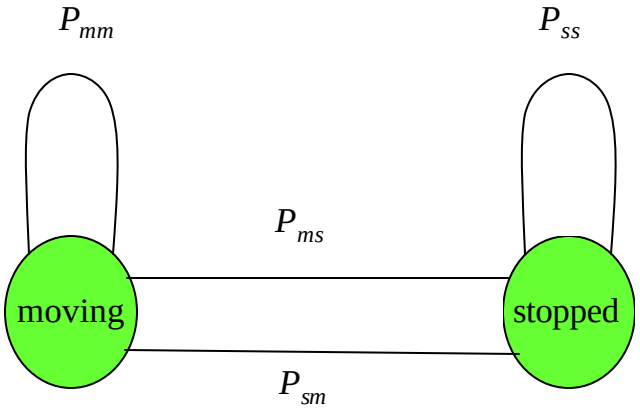


Figure 2. Discrete-time Markov chain modeling target motion. At any time, either a target is moving or it is stopped. The transitions occur at times $kd$, for $k$ = 1,2,…, where $d$ is the time increment.

We consider a 10 minute scenario in which there are 50 targets to be tracked while they are both moving and stopped by a radar having two modes, MTI and FTI. We assume that each target moves along a one-dimensional road and has normally distributed velocity with a specified root mean square value. The target transitions between being moving and stopped are modeled by a discrete-time Markov chain with two states, moving and stopped, and with state dependent transition probabilities, as illustrated in Figure 2. By varying $P_{sm}$, we simulate the cases where the average number of targets that are stopped in steady-state is 20, 30, and 40, and we initiate the number of targets being stopped in the scenario to the average steady-state value.

Our tracker model is based on a simple IMM filter (cf. [8]) consisting of a filter modeling the kinematics of a "moving" target and another one modeling the kinematics of a "stopped" target. The target track state consists of the target location estimate, the estimate of error variance, and the target mode probabilities. The target mode probabilities are updated given information on whether a target is detected or not. In particular, if a target is not detected in MTI mode, then the probability that the target is moving decreases. A similar update results from a missed detection in FTI mode. The tracker drops a target if its mean-squared error exceeds a specified upper bound. Furthermore, the tracker report association is assumed to be perfect.

The MTI radar mode can detect moving targets only, while the FTI mode can detect stationary targets only. Each mode is characterized by its detection probability,

measurement error variance, and dwell-time. For both modes and all targets, the detection probability is 0.9 and the measurement error variance is 1 $m^2$. The MTI mode dwell-time is 0.1 seconds, and the FTI mode dwell-time is 10 seconds.

We assume that all targets have the same priority and the same goal values for their error variances. In particular, in equation (1), we use priority $V_i$=1 and goal value $G_i$=25 $m^2$ for all $i$.

### 4.2 Simulation results

In this section, we present the simulation results obtained for the farsighted sensor management algorithm and the myopic entropy-optimizing algorithm. We use the average fraction of time the error goals are met as a measure of performance. This is computed as the fraction of time the target error goal is met per target and then averaged over the number of targets. These average values, obtained for typical sample paths, are presented in Figure 3. The bars in the figure mark the standard deviations of performance over time for each sample path. They give an indication of the expected variability in performance for different sample paths provided that the dynamics are ergodic.

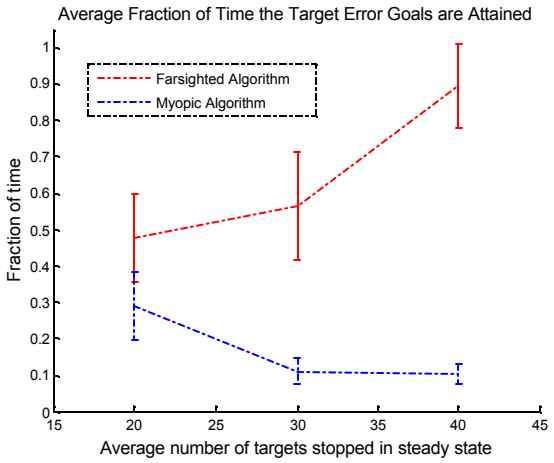


Figure 3. As the average number of stopped targets increases, the average time the error goals are met for the farsighted algorithm is increasingly better than that of the myopic algorithm.

The simulation results indicate that the farsighted sensor management algorithm maintains better quality target tracks than the myopic algorithm. In particular, the average time the error goals are met for the farsighted algorithm is much longer than for the myopic algorithm. Furthermore, the difference between the average time the error goals are met for the farsighted algorithm and for the myopic algorithms is increasing as the average number of stopped targets increases. We attribute this to the capability of the farsighted algorithm to adapt the target revisit rates appropriately.

In particular, the myopic algorithm schedules the longer FTI mode more frequently to observe the stopped targets, as the number of stopped targets increases. This results in substantially less time being spent observing moving targets, and the corresponding track errors far exceed the goals. In contrast to the myopic algorithm, the farsighted algorithm schedules the FTI mode less frequently. Instead, it schedules shorter MTI modes to revisit stopped targets at an appropriate rate to determine if they have started moving. The resulting revisit rates on the moving targets are faster, and the farsighted algorithm is able to maintain track errors on the moving targets below the desired goals for longer periods of time.

We believe that our simulation results are important indicators that, for some sensor management problems, farsighted strategies are better than myopic ones. We believe that this is the case for sensor management problems with complex dynamics (e.g., when targets are randomly starting and stopping and/or sensor actions have significantly different durations). The move/stop tracking problem considered here is one such. For these problems, the consequences of a single sensor action do not provide enough information about the impact of the action on the future system behavior. Thus, to make good decisions, it is important that the sensor manager anticipates future consequences resulting from the sensor actions taken at the present time. In particular, the farsighted, rollout sensor manager results in a strategy that adaptively adjusts the frequency with which moving and stopped targets are observed in a manner that results in better tracks than the myopic, entropy-optimizing algorithm.

## 5 Conclusions

We have developed a novel, computable, farsighted sensor manager for move/stop tracking with a multi-mode sensor. This particular sensor management problem is challenging because of the complex target dynamics and the variable duration of sensor actions. We have evaluated the farsighted algorithm against a myopic, entropy-optimizing sensor management algorithm. Our simulation results indicate that the farsighted algorithm has promising behavior. In particular, the farsighted algorithm results in better quality tracks than the myopic algorithm.

## Acknowledgements

This material is based upon work supported by the United States Air Force under Contract No. F33615-02-C-1197.

# References


[1] Kreucher C., Kastella K., and Hero A.O., *Sensor Management Using Relevance Feedback Learning*. Submitted to IEEE Trans. on Signal Processing.

[2] Ertin E.J., Fisher W., and Potter L.C. *Maximum Mutual Information Principle for Dynamic Sensor Query Problems*. Information Processing in Sensor Networks, Proc. pp. 405-416, 2003.

[3] Zhao F., Shin J., and Reich J. *Information-driven dynamic sensor collaboration*. IEEE Signal Processing Magazine, Vol. 19, pp. 61-72, 2002.

[4] Krishnamurthy V. and Evans R.J. *Hidden Markov model multiarm bandits: A methodology for beam scheduling in multitarget tracking,* IEEE Transactions on Signal Processing, Vol. 49, No. 12, pp. 2893-2908, Dec. 2001.

[5] Washburn R.B., Schneider M.K., and Fox J.J. *Stochastic Dynamic Programming Based Approaches to Sensor Resource Management*. Fifth International Conference on Information Fusion, Proc. pp. 608- 615, 2002.

[6] Kreucher C., Hero A.O., Kastella K., and Chang D., *Efficient Methods of Non-myopic Sensor Management for Multitarget Tracking*. Submitted to Proceedings of the 2nd IEEE Conference on Information Processing in Sensor Networks, 2004.

[7] Bertsekas D.P., Dynamic Programming and Optimal Control. Athena Scientific, Belmont, Vol. 1 and 2, Second edition, 2000.

[8] Mazor E., Averbuch A., Bar-Shalom Y., and Dayan J. *Interacting multiple model methods in target tracking: a survey*. IEEE Transactions on Aerospace and Electronic Systems , Vol. 34, No. 1, pp.103-123, 1998.

[9] Cover T.M. and Thomas J.A. *Elements of information theory*. John Wiley & Sons, New York, 1991.

# Stochastic Control Bounds on Sensor Network Performance

David A. Castañón

*Abstract*— **Consider a network of sensors, each of which has limited sensing resources, which is tasked with collecting noisy classification information on a group of unknown objects. The amount of resources required a given sensor to measure an object depends on the specific sensor-object geometry. Sensors exchange collected information to estimate object identities and coordinate which measurements to collect next. This paper describes a computable lower bound on the classification error that can be achieved by a causal adaptive sensing schedule. This bound is based on a formulation of the adaptive sensing problem as a partially observed stochastic control problem. Expanding the admissible control space of this problem leads to a relaxed problem with simpler decision structure for which the bounds can be computed. The bound computations are illustrated for several examples involving 100 unknown objects, and compared with the Monte Carlo performance of specific adaptive sensor scheduling algorithms. Comparisons with optimal scheduling algorithms for special cases illustrate the tightness of the bounds.**

## I. INTRODUCTION

There are many recent applications for networks of sensors, each of which has a given amount of resources, such as available power or duty cycle. Often, each sensor has multiple sensing modes that it can use to collect different types of information; the amount of resources required to collect a measurement by a sensor depends on the specific sensor-object geometry and the mode used. The network is tasked with using its available resources to obtain information on a given number of objects or areas. In order to achieve the best information possible, it is important to coordinate the allocation and scheduling of the different sensors and sensor modes across objects of interest. Sensors exchange collected information to determine the current state of information on objects. The adaptive sensing problem consists of selecting and scheduling the sensor modes which are applied to objects of interest based on the collected past information.

This paper develops a model for a class of adaptive sensing problems involving the objective of classifying a known number of unknown objects at known locations, given a fixed number of sensor with finite resources and finite modes. We assume that sensor performance parameters are time-invariant, so that the performance associated with a sensor observing an object with a given mode do not depend on the time that the sensing activity occurs. This class of problems arises in several applications, from object classification using multiple airborne platforms, dynamic search, and fault inspection and isolation in manufacturing systems. In these applications, inaccuracies in sensor measurements and variations in object characteristics result in individual measurements that provide noisy estimates of object type whose quality depends on the specific mode used by the sensor. In situations with multiple sensors, multiple objects and limited resources, this noisy information can be used to prioritize which objects to look at next, from which sensor, and to assign appropriate sensor modes to the objects.

Because of the uncertain nature of the underlying object types and the adaptive nature of the desired schedules, adaptive sensing problems can be formulated as partially observed Markov decision problems (POMDP) [1], [2], [10], [11]. As such, this class of problems can be solved using stochastic dynamic programming [3]. However, for large numbers of objects, the required state space is very high-dimensional, consisting of the conditional probability distributions of all of the objects. This leads to intractable computational problems, even with the fastest POMDP algorithms.

Sensing problems have been formulated previously as dynamic optimization problems with partial information. The extensive literature in search theory [20] deals with sensor management problems involving objects that can be of one of two types (hidden or found) with sensors that have only a single mode. The dynamic hypothesis testing problems studied in [6] also have objects that can be of two types and a single sensor mode, but generalize results in search theory to broader classes of measurements. More recently, there has been work [17] using Markov decision problem techniques for sensor management, particularly techniques based on the solution of multiarmed bandit problems. However, these formulations also restrict the sensors to a single sensor with a single mode, and require an infinite horizon, time-invariant formulation.

Because of the complexity of general adaptive sensing algorithms with multiple sensors and modes, most practical algorithms are heuristic algorithms based on information-theoretic metrics [5]. To date, there has been no effective approach that can characterize the achievable adaptve sensing performance performance to determine whether such heuristic algorithms are performing well.

In this paper, we consider sensing problems involving multiple distributed sensors with multiple modes per sensor. This model is an extension of the model discussed in [7]. We show that the resulting POMDP models admit a lower bound on classification error performance based on modifying the constraint structure to expand the space of admissible strategies. The resulting problem becomes a dynamic optimization problem subject to expected value constraints, a class of problems recently studied by in [24]. We develop a hier-

This work was supported in part by grants NSF DMI-0330171 and DARPA F33615-02-C-1197

Dept. Electrical & Computer Eng., Boston University, Boston, MA dac@bu.edu

archical algorithm that exploits the structure of the resulting relaxed problem. This hierarchical algorithm is based on the solution of single object POMDP problems, coupled with nondifferentiable optimization techniques based on Lagrangian relaxation [16]. The single object problems are of small dimension, and can be readily solved using standard algorithms for POMDPs [10], [11], [13]. The hierarchical algorithm avoids the exponential growth of the dimensions of the resulting state space in the POMDP problem as a function of the number of objects.

The paper includes several examples where the lower bound performance is computed, and compared with the Monte Carlo performance achieved by suboptimal SM algorithms. In particular, we compute bounds for a special problem for which the optimal sensing strategy is known, and compare the bounds to the optimal performance to show how tight the bounds are.

## II. PROBLEM FORMULATION

In this section, we develop a formulation of the adaptive sensing problem as a partially observed Markov decision problem (POMDP). Assume that there are $N$ objects of interest in the problem, with known locations. Each object can belong to one and only one of $K$ different classes, and the object identity does not change over time. Let the variable $x_i \in X \equiv \{1, \ldots, K\}$ denote the true class of object $i$. We define the complete (but unknown) system state as:

$$\underline{x} = \begin{pmatrix} x_1 & x_2 & \cdots & x_N \end{pmatrix} \tag{1}$$

Since the identities do not change over time, the complete system state is constant over time. We assume that $x_i$ are independent random variables with values in the finite space $X$. Associated with each object $i$ is a prior probability vector $\pi_i(0)$ which describes the probability distribution of the random variable $x_i$. That is,

$$\pi_{ij}(0) = \text{Prob}\{x_i = j\} \tag{2}$$

These probability distributions represent a priori knowledge collected on each object.

To obtain information about the state of each object, selected objects are examined with different modes from different sensors. In order to simplify the notation in the exposition, we consider the case of a single sensor with multiple modes $m \in \{1, \ldots, M\}$. We will highlight later the extensions required to incorporate multiple sensors. The action to use a sensor mode $m$ on object $i$ produces an observable $y_m$ in a finite set $Y_m$, with a conditional probability distribution that depends only on the object $i$, its type $x_i$ and the mode $m$, denoted by $p(y_m|i, x_i, m)$. We assume that the observation outcomes of these sensing actions are conditionally independent of each other given the object types.

We assume that obtaining a measurement of object $i$ with mode $m$ requires sensor resources $R_{im} > 0$ (e.g. power), which depend on the object location, sensor location and specific mode selected. The sensor has a finite amount of sensor resources $R$ that can be used for measuring objects. The objective is to classify, with minimal error cost, the objects after the sensor resource $R$ is exhausted. This formulation is stated below.

Without loss of generality, we restrict our attention to sensing policies that execute only one action at a time. Such strategies are optimal in that they provide maximal information for adaptation, and will achieve minimal error cost. Let $u(k) = (i(k), m(k))$ denote the $k+1$-th action (starting at $k = 0$) taken by the sensor, consisting of measuring object $i(k)$ with mode $m(k)$. Let $U$ denote the set of possible sensor actions, and let $y_{m(k)}(k)$ denote the measured value resulting from action $u(k) \in U$. The past information available to adaptively select $u(k)$ is $I(k) = \{u(0), y_{m(0)}(0), \ldots, u(k-1), y_{m(k-1)}(k-1)\}$. The sensing problem decisions are selected adaptively until a final random stopping instance $T$, selected based on the information $I(T)$. At the end of this stopping instance, the information $I(T)$ is available for estimating the object types. For each object $i$, there is a final decision $v_i \in X$ based on $I(T)$ that is selected to minimize the expected classification error.

An admissible adaptive sensing policy is a set of measurable feedback policies $\{\gamma(0), \ldots, \gamma(T)\}$ and stopping time $T$ such that

$$\begin{aligned} &\gamma(k) : I(k) \to U, \ k < T \\ &T : I(T) \to \{stop, continue\} \\ &\gamma(T) : I(T) \to X^N \end{aligned} \tag{3}$$

Let $\Gamma$ denote the set of all admissible sensing policies. Since the observation space is finite and the decision space is also finite, $\Gamma$ is a countable space.

Denote by $c(v, x)$ the cost of selecting classification decision $v$ when the true object type is $x$. The adaptive sensing problem is to minimize the expected total classification cost

$$J(\gamma) = E_\gamma\{\sum_{i=1}^{M} c(v_i, x_i)\} \tag{4}$$

over adaptive sensing policies $\gamma \in \Gamma$ satisfying the resource utilization constraint

$$\sum_{k=0}^{T-1} R(u(k)) \leq R \tag{5}$$

with the notation $R(u(k)) \equiv R_{i(k)m(k)}$. Note that the constraint in (5) is a sample path constraint; for every realization of the information sets $I(k)$, the adaptive policy $\gamma$ must not exceed the total sensor resources available. Since the sets of possible observation outcomes per mode $Y_m$ and possible decisions $u_m$ are finite, the number of possible information sets after $k-1$ actions $I(k)$ is also finite. This implies that there is a finite number of possible admissible sensing policies that satisfy the constraint (5).

The above problem is a class of finite-state, finite-observation partially observed Markov decision problems studied in [1], [2], [10], [11], with the special structure that the underlying state dynamics are trivial, and the presence of the sample path constraints of (5). Such problem

scan be transformed into fully-observed Markovian decision problems in terms of a sufficient statistic: the conditional probability distribution of the state $\underline{x}$ given information $I(k)$, as follows: Let $S \subset R^K$ denote the space of probability distributions on $X$, and let $S_N$ denote the space of probability distributions on $X^N$. The conditional distribution vector for the composite state $\underline{x}$ given the information $I(k)$, $P(\underline{x}|I(k)) \in S_N$, can be viewed as an information state, a sufficient statistic summarizing the past observations. The recursive evolution of this information state in response to an action $u(k) = (i(k), m(k))$ can be described by Bayes' rule as

$$\begin{aligned} P(\underline{x}|I(k+1)) &= P(\underline{x}|I(k), u(k), y_{m(k)}(k)) \\ &= \frac{P(y_{m(k)}(k)|x_{i(k)}, m(k))P(\underline{x}|I(k))}{P(y_{m(k)}(k))|I(k), u(k))} \end{aligned} \quad (6)$$

with the initial condition

$$P(\underline{x}|I(0)) = \prod_{i=1}^{N} \pi_i(0) \quad (7)$$

Under the previous independence assumptions, the following lemma establishes a convenient representation:

*Lemma 2.1:* Under the adaptive sensing problem assumptions, the conditional probability $P(\underline{x}|I(k))$ can be factored as

$$P(\underline{x}|I(k)) = \prod_{i=1}^{N} P(x_i|I(k)) \quad (8)$$

where the evolution of $P(x_i|I(k))$ under sensing action $u(k) = (i(k), m(k))$ and observation $y_{m(k)}(k)$ is given by

$$P(x_i|I(k+1)) = \begin{cases} P(x_i|I(k)) & \text{if } i(k) \neq i \\ \frac{P(y_{m(k)}(k)|x_{i(k)}, m(k))P(x_i|I(k))}{\sum_{j=1}^{K} P(y_{m(k)}(k)|x_i=j, I(k))P(x_i=j|I(k))} & \text{otherwise} \end{cases} \quad (9)$$

The proof of this lemma is straightforward by induction, as the independence assumption of the object types $x_i$ guarantees the Lemma is satisfied at $k = 0$, and (6) establishes the recursion. Note also that $P(x_i|I(k))$ depends only on measurements in $I(k)$ corresponding to object $i$.

Define $\pi_i(k)$ to be the conditional probability distribution of $x_i$ given information $I(k)$:

$$\pi_i(k) = P(x_i|I(k)) \quad (10)$$

The vector $\pi_i(k)$ has components $\pi_{ij}(k) = P(x_i = j|I(k))$. Lemma 2.1 establishes that the conditional probability distribution of the entire state, $P(\underline{x}|I(k))$, can be computed as the product of $\pi_i(k), i = 1, \ldots, N$. Define the information vector $\vec{\pi} = \begin{pmatrix} \pi_1^T & \ldots & \pi_N^T \end{pmatrix}^T$ For a given observation $y_m$ using mode $m$ on object index $i$, define the observation probability matrix as the $K \times K$ diagonal matrix

$$B_i(y_m) = \text{diag}\{P(y_m|x_i = 1, m), \ldots, P(y_m|x_i = K, m)\}$$

The information vector evolves in response to a measurement $y_m$ obtained from a sensing action $(i, m)$ according to an operator $T$, where

$$T(\vec{\pi}, u = (i', m), y_m) = \begin{pmatrix} T_1(\pi_1, u = (i', m), y_m) \\ \vdots \\ T_n(\pi_N, u = (i', m), y_m) \end{pmatrix}$$

and

$$T_i(\pi_i, u = (i', m), y_m) = \begin{cases} \pi_i & \text{if } i \neq i' \\ \frac{B_i(y_m)\pi_i}{\vec{e}^T B_i(y_m)\pi_i} & \text{if } i = i' \end{cases}$$

and $\vec{e}$ is a $K$-dimensional vector of all ones.

The adaptive sensing problem described above can be solved by stochastic dynamic programming [3]. The resource constraint in (5) can be incorporated into the dynamics to obtain a dynamic programming recursion, as follows [24]. Define a value function $V(\vec{\pi}, C)$ to be the optimal solution of (3)-(5) when the initial information is $\vec{\pi}$ and the available sensor resource level is $R = C$. The value function $V$ is thus defined on $S^N \times R^+$. The dynamic programming problem is stated as a total cost problem with nonnegative costs, for which the optimal value function satisfies the following Bellman's equation: Let $U(R) \subset U$ denote the set of feasible sensor actions $(i, m)$ such that $R_{im} \leq R$. At each decision stage, there is a choice of stopping and classifying the objects with the available information, or taking additional measurements. The optimal value function satisfies the Bellman equation

$$\begin{aligned} V(\vec{\pi}, R) = \min\Big[&\sum_{i=1}^{N} \min_{v_i \in X} \sum_{j=1,\ldots,K} c(v_i, j)\pi_{ij}, \\ &\min_{u \equiv (i',m') \in U(R)} E_{y_{m'}}\{V(T(\vec{\pi}, (i', m'), y_{m'}), R - R_{i'm'})\}\Big] \end{aligned} \quad (11)$$

where

$$\begin{aligned} E_{y_{m'}}&\{V(T(\vec{\pi}, (i', m'), y_{m'}), R - R_{i'm'})\} = \\ &\sum_{y_{m'} \in Y_{m'}} P(y_{m'}|I(k), u)V(T(\vec{\pi}, u, y_{m'}), R - R_{i'm'}) \\ &= \sum_{y_{m'} \in Y_{m'}} e^T B_{i'}(y_{m'})\pi_i V(T(\vec{\pi}, u, y_{m'}), R - R_{i'm'}) \end{aligned} \quad (12)$$

This recursion starts from the following boundary conditions: Let $R_{min} = \min_{i,m} R_{im}$. Then, the set of admissible modes $U(R)$ is empty for $R < R_{min}$. Thus,

$$V(\vec{\pi}, R) = \sum_{i=1}^{N} \min_{v_i \in X} \sum_{j=1,\ldots,K} c(v_i, j)\pi_{ij} \quad \text{if } R < R_{min} \quad (13)$$

Eqs. (11)-(13) can be used recursively to compute the optimal value for all information states and nonnegative levels.

The initialization of the recursion decouples into $N$ independent optimizations, as there are no coupling constraints on the decisions $v_i$, and the local decision costs $c(v_i, x_i)$ depend only on the marginal probability distributions of each object's type. However, the recursion (11) does not preserve this decomposability. The coupling arises primarily

because of the resource use constraints in (5); the decision of which object to view and which mode to use depends on the information vector of all the objects and the available resources. Thus, the dynamic programming induction must be carried out for the entire state $\vec{\pi}(t)$, which becomes a formidable computation problem even for moderate numbers of objects.

## III. RELAXED FORMULATION AND LOWER BOUNDS

To obtain a simpler dynamic programming formulation, we relax the sample path sensor resource use constraints (5) and use an averaged version of the same constraints, as

$$E\{\sum_{k=1}^{T} R(u(k))\} \leq R \tag{14}$$

This approach replaces a large set of constraints (one per sample path) by a single aggregate constraint. Note that any adaptive sensing policies that satisfy (5) will also satisfy (14). Thus, this approach increases the set of admissible policies. Let $J^*$ denote the optimal classification cost of the problem in (3)-(4) with constraints (5). Let $J_A^*$ denote the optimal classification cost of the problem in (3)-(4) with constraints (14), denoted as the relaxed adaptive sensing problem.

*Lemma 3.1:* $J^* \geq J_A^*$

The relaxed problem has a single coupling constraint (for one sensor) relating the sensing actions on different objects. Let $\lambda \geq 0$ denote a Lagrange multiplier. For any admissible policies in $\Gamma$, consider the objective

$$J(\lambda, \gamma) = E_\gamma\{\sum_{i=1}^{N} c(v_i, x_i)\} + \lambda[E_\gamma\{\sum_{k=0}^{T-1} R(u(k))\} - R] \tag{15}$$

Consider the unconstrained adaptive sensing problem of finding policies $\gamma$ and an adaptive stopping time $T$ to minimize (15). If $(\gamma, T)$ is an adaptive SM policy that satisfies (14), the second term in (15) is nonpositive. Denote by $J^*(\lambda)$ the optimal value of (15) over all adaptive sensing policies $\gamma \in \Gamma$. Then,

*Lemma 3.2:* For all values of $\lambda \geq 0$,

$$J^* \geq J_A^* \geq J^*(\lambda) \tag{16}$$

In particular,

$$J^* \geq \sup_{\lambda \geq 0} J^*(\lambda) \tag{17}$$

Lemma 3.2 is a consequence of weak duality in nonlinear programming [4]. Note that the number of adaptive sensing policies that satisfy (15) is finite, because the set of possible histories $I(k)$ is finite for all $k$. Thus, computation of $J_A^*$ is an integer programming problem, and computation of $\sup_{\lambda \geq 0} J^*(\lambda)$ is its dual problem.

The key issue is whether the lower bounds $J^*(\lambda)$ can be computed efficiently. Rewrite (15) for $\gamma \in \Gamma$ as

$$J(\lambda, \gamma) = E_\gamma\{\sum_{i=1}^{N}[c(v_i, x_i) + \lambda \sum_{k=0}^{T-1} R(u(k))\delta_{i(k)-i}]\} - \lambda R \tag{18}$$

where the indicator function $\delta_i = 1$ if $i = 0$, and 0 otherwise. This suggests that optimization of $J(\lambda, \gamma)$ may be separable across individual objects $i$.

Partition the information $I(k)$ into disjoint sets $I_i(k)$, where $I_i(k)$ are the sensing actions and measurement actions applied to object $i$:

$$I_i(k) = \{(u(j), y(j)) | j < k, i(j) = i\} \tag{19}$$

Note that the conditional probability vector $\pi_i$ only changes on measurements included in $I_i(k)$. We wish to restrict the set of adaptive sensing policies to a subset where the decision to apply a sensor action for object $i$ depends only on the information previously collected for object $i$. We refer to this subset of policies as adaptive *local* sensing policies, defined as:

*Definition 3.1:* An adaptive *local* sensing policy is an adaptive sensing policy $\gamma$ and stopping times $T_i, i = 1, \ldots, N$, with the properties that, for each sensing action instance $k$,

1) If $u(k) = (i(k), m(k))$, then $i(k) = k \mod N + 1$.
2) The selected sensor mode $m(k)$ depends only on the information $I_{i(k)}$.
3) For each object $i$, there is a stopping time $T_i$ which depends only on $I_i(T_i)$ such that, for all $k \geq T_i$, if $i = k \mod N + 1$, no sensing action is taken. If $k < T_i$ and $i = k \mod N + 1$, then $u(k) = (i, m)$ for some mode $m$ in {1, …, M}.
4) At time $T_i$, the local decision $v_i$ for object $i$ is selected as a function of $I_i(T_i)$.

Adaptive local sensing policies use a fixed round-robin schedule for selecting which objects to measure. Furthermore, the choice of sensing mode for each action on object $i$ depends only on the prior information collected on that object. In addition, there is an independent stopping time for each object $i$ such that a final classification decision is made on object $i$, based only on prior information collected on that object. Note that there are decision instances $k$ where no sensing action is taken, when $k \geq T_i$ and $i = k \mod N + 1$; these instances correspond to times after a final decision has been selected for object $i$. The *effective* stopping time of an adaptive local sensing policy is defined as $T = \max_{i=1,\ldots,N} T_i$, and is the earliest time at which every object has a final classification decision. Thus, adaptive local sensing policies can be viewed as a subset of the class of adaptive sensing policies.

Let $\Gamma_L$ denote the set of adaptive local sensing policies. For a given amount of sensor resources $R$, there are a finite number of feasible adaptive local sensing policies. In general, $\Gamma_L$ is a countable discrete set. For the purposes of bound computation, we will expand $\Gamma_L$ to include *mixed* policies, consisting of probabilistic mixtures of policies in $\Gamma_L$:

*Definition 3.2:* A *mixed* local sensing policy is a probability distribution $q(\gamma)$ over $\Gamma_L$ such that local SM policy $\gamma$ is selected for use with probability $p(\gamma)$. The set of mixed local sensing policies is denoted by $Q(\Gamma_s)$.

Consider the problem of minimizing the relaxed cost (18) over local sensing policies $\Gamma_L$. Since $\Gamma_L \subset \Gamma$, we have

$$\min_{\gamma\in\Gamma} J(\lambda,\gamma) \leq \min_{\gamma\in\Gamma_L} J(\lambda,\gamma) \tag{20}$$

Furthermore, since (18) is an unconstrained objective, the minimum in mixed local sensing policies is achieved by a pure local sensing policy, so

$$\min_{\gamma\in\Gamma} J(\lambda,\gamma) \leq \min_{q\in Q(\Gamma_L)} \sum_{\gamma\in\Gamma_L} q(\gamma) J(\lambda,\gamma) \tag{21}$$

The importance of mixed local sensing policies is highlighted in the theorem below, proven in [25]:

*Theorem 3.1:* For any admissible adaptive sensing policy $\gamma \in \Gamma$, there exists a mixed local sensing policy $q \in Q(\Gamma_s)$ such that the expected classification costs in (4) and the expected total resource use in (14) are equal under both policies $\gamma$ and $q$.

This result implies the following inequality:

$$\min_{\gamma\in\Gamma} J(\lambda,\gamma) \geq \min_{q\in Q(\Gamma_L)} \sum_{\gamma\in\Gamma_L} q(\gamma) J(\lambda,\gamma) \tag{22}$$

Combining (21) and (22) yields the following:

$$\min_{\gamma\in\Gamma} J(\lambda,\gamma) = \min_{q\in Q(\Gamma_L)} \sum_{\gamma\in\Gamma_L} q(\gamma) J(\lambda,\gamma) = \min_{\gamma\in\Gamma_L} J(\lambda,\gamma) \tag{23}$$

Eq. (23) implies that lower bounds for the achievable classification performance can be computed by optimizing over local sensing policies only. For each local policy $\gamma \in \Gamma_L$, let $\gamma_i$ denote the policy that is used for instances $k$ when actions are taken for object $i$, and let $\Gamma_{L_i}$ be the set of such admissible local policies for object $i$. Thus, $\gamma_i$ selects actions for object $i$ based on past observations $I_i(k)$, and selects a stopping time $T_i$ and a final classification $v_i$ at that stopping time. The importance of local sensing policies is that the optimization in (23) decouples over objects as

$$\min_{\gamma\in\Gamma_L} J(\lambda,\gamma) = \sum_i \min_{\gamma_i\in\Gamma_{L_i}} J_i(\lambda,\gamma_i) - \lambda R \tag{24}$$

where

$$J_i(\lambda,\gamma_i) = E_{\gamma_i}\{c(v_i,x_i) + \lambda \sum_{k=0}^{T-1} R(u(k))\delta_{i(k)-i}\} \tag{25}$$

This implies that computation of the bounds can be achieved with $N$ independent optimization problems for each value of $\lambda$. Furthermore, the optimal bound can be computeed as in Lemma 3.2, as

$$J^* \geq \sup_{\lambda\geq 0} \min_{\gamma\in\Gamma_L} J(\lambda,\gamma) \tag{26}$$

Note that the right hand side of (26) is the dual of the following linear programming problem:

$$\min_{q\in Q(\Gamma_L)} \sum_{\gamma\in\Gamma_L} q(\gamma) E_\gamma J(\gamma) \tag{27}$$

subject to

$$\sum_{\gamma\in\Gamma_L} q(\gamma) E_\gamma[\sum_{k=0^{T-1}} R(u(k))] \leq R \tag{28}$$

$$\sum_{\gamma\in\Gamma_L} q(\gamma) = 1 \tag{29}$$

which is a linear program over the choice of probability distributions $q \in Q(\Gamma_L)$. This can be exploited to solve efficiently for the bound. Specifically, note that this is a linear program subject to two constraints, which implies that the optimal mixed local sensing policy $q$ will have support only on two pure local sensing policies. This property will be exploited in the next section for bound computation.

## IV. BOUND ALGORITHMS

There are two potential approaches to compute a lower bound: a dual approach, based on Lagrangian relaxation [16], that optimizes (26) over the choice of dual variable $\lambda$, and a primal approach based on solving the linear program (27)-(29). The dual approach is straightforward, and uses techniques from nondifferentiable optimization [19] to search the space of possible $\lambda$. The primal approach is harder, because the optimization is over a large space of possible values of mixture probabilities $q$. However, this mixture has very sparse support, which makes it suitable for column generation algorithms [18].

A fundamental step in either approach is the computation of the optimal local sensing policies for a fixed value of $\lambda$ for each object. For object $i$, one must solve the local problem given $\lambda$:

$$\min_{\gamma_i} E_{\gamma_i}[c(v_i,x_i) + \lambda \sum_{k\geq 0: i=k \bmod N+1}^{T_i-1} R(u(k))] \tag{30}$$

This problem is a multi-stage single object POMDP, with sufficient statistic given by the marginal probability distribution $\pi_i(k)$. One can reduce the action instants to a new counter $k'$ indexing only the action opportunities for object $i$, to obtain

$$\min_{\gamma_i\in\Gamma_{Li}} E_{\gamma_i}[c(v_i,x_i) + \lambda \sum_{k'}^{T_i'-1} R_{im(k')}] \tag{31}$$

The resulting POMDP problems are small enough to solve using existing algorithms such as those overviewed in [2], [10], [13], [14].

Solution of the $N$ decoupled problems (31) yields a local policy $\gamma \in \Gamma_L$, for which the expected classification cost $E_\gamma[\sum_{i=1}^N c(v_i,x_i)]$ and expected resource use $E_\gamma[\sum_{k=0}^{T-1} R(u(k))]$ are computed from the solution. This provides the starting point for the use of column generation [18] for solution of (27)-(29). Column generation was used by Yost [21], [22], [23] in his work on POMDPs for resource assignment and was also exploited in [8] for the solution of stochastic weapon assignment problems.

The algorithm starts with an initial set of pure local sensing policies $\gamma^d$ indexed by $d = 1,\ldots,D$, with known expected classification performance $J^d$ and expected resource

use $R^d$. The first step in the algorithm is to solve the linear program in (27)-(29) restricted to mixtures of the $d = 1, \ldots, D$ initial policies. Since the support of the admissible mixed policies is restricted, the solution provides an upper bound $J^{UB}$ to the optimal cost. Denote by $\lambda_D$ the optimal dual price of the resource constraint (29) in this solution. The constraint generation algorithm uses this optimal dual price value in (31) to generate a new candidate local policy $\gamma^{D+1}$,by solving $N$ independent POMDP problems with this value of $\lambda$. The combined solution of the $N$ subproblems also provides a lower bound $J^{LB}$ on the optimal performance, as described in Lemma 3.2. The key result in the constraint generation algorithm is stated as follows [18]:

*Lemma 4.1:* Consider the pure local policy generated by the solution of (31). If $J^{LB} = J^{UB}$, the optimal solution over all mixtures of local policies is a mixture of the local policies indexed by $d = 1, \ldots, D$. Otherwise, the pure local policy $\gamma^{D+1}$ can be used as part of a mixed policy which provides a cost lower than $J^{UB}$.

## V. EXTENSION TO MULTIPLE SENSORS

The development of the previous sections carries through with little modification when multiple sensors are used. The key difference is that there is a separate resource constraint for each sensor. Thus, there will be a vector of sensor resources $R_s$, where $s$ is a sensor index, thus resulting in a vector of averaged constraints (14). The Lagrange multipliers $\lambda$ will thus be vectors instead of scalars. Nevertheless, all of the lemmas and theorems can be extended to the multisensor case with minor modifications.

The main assumption that was used in the single sensor formulation was that only one sensor action would be performed simultaneously. This assumption is still used for the multiple sensor problem to derive the lower bound, although the results in the previous section indicate that optimal local sensing strategies that achieve the lower bound may use simultaneous sensing by multiple sensors.

The column generation algorithm discussed in the previous section extends naturally to multiple sensors. When there are $L$ sensors, the optimal mixed local SM policies will be mixtures of $L+1$ pure local SM policies. Nondifferentiable optimization algorithms that maximize the dual cost can also be used in this case.

## VI. EXAMPLES

For our first example, we consider a case where the optimal strategies are known [26]. In this example, there are 100 unknown objects with one of two types, with equal priors for each object. There is a single sensor that has a single measurement mode, and the problem is optimal adaptive allocation of a fixed number of measurements over the number of objects. Measurement outcomes are binary-valued, identifying one of the two types, and a single measurement has a probability of error $P_e$, which is symmetric over type. The objective is to minimize the expected number of classification errors after $N$ measurements. The optimal strategy derived in [26] is to assign the next measurement to the object with conditional probability.

Table I shows the results of 1000 Monte Carlo simulations of the optimal strategy, compared with the predicted performance of the lower bound, in terms of expected number of classification errors for 3 different conditions of symmetric single measurement $P_e$ and four levels of number of measurements $N$. As the table indicates, the bound predictions are very tight for this case. The gap between gap and optimal strategy increases as the number of measurements $N$ increases because likelihood of errors decreases, and the bound strategy allows the use of more resources than available in the unlikely cases that lead to errors.

| | $P_e = 0.25$ | | $P_e = 0.2$ | | $P_e = 0.15$ | |
|---|---|---|---|---|---|---|
| N | Bound | Opt. | Bound | Opt. | Bound | Opt. |
| 100 | 25 | 25.03 | 20 | 20.02 | 15 | 15.067 |
| 200 | 18.182 | 18.185 | 12.727 | 12.765 | 7.888 | 7.988 |
| 300 | 11.364 | 11.432 | 5.749 | 6.038 | 2.518 | 2.593 |
| 400 | 7.833 | 7.905 | 3.468 | 3.543 | 0.927 | 0.987 |

TABLE I

COMPARISON OF EXPECTED NUMBER OF ERRORS BY LOWER BOUND AND MONTE CARLO OF OPTIMAL STRATEGY

For the second set of experiments, we consider a different 100 object scenario where objects can be of three different types ($K = 3$): cars, trucks and military vehicles (MV). There is a single sensor, with two modes: a low resolution mode 1 that takes 1 second per image ($R_{i1} = 1$), and a high resolution mode 2 that requires 5 seconds per image, ($R_{i2} = 5$). Low resolution imagery is useful in separating cars from trucks and MVs, but separating trucks from MVs requires high resolution imagery. Apriori, each object has a probability of 0.1 as a military vehicle, 0.2 truck and 0.7 car. Imagery generated by the sensor is processed into a binary decision as to whether the object is MV or not. Hence $y_{ij} \in \{0, 1\}$, where 1 indicates that the decision is MV.

The objective of the problem is to determine as accurately as possible which objects are military vehicles (type 1). Thus, the classification costs are given by $d(v_i, x_i)$ as a $3 \times 3$ matrix where $v_i$ is the row index:

$$(d(v_i, x_i)) = \begin{pmatrix} 0 & MD & MD \\ FA & 0 & 0 \\ FA & 0 & 0 \end{pmatrix} \tag{32}$$

where $FA = 1$ and $MD$ will vary from 1 to 80 in the experiments.

| | low-resolution | | high-resolution | |
|---|---|---|---|---|
| Type | $y = 0$ | $y = 1$ | $y = 0$ | $y = 1$ |
| Car | 0.9 | 0.1 | 0.95 | 0.05 |
| Truck | 0.1 | 0.9 | 0.85 | 0.15 |
| MV | 0.1 | 0.9 | 0.8 | 0.2 |

TABLE II

MEASUREMENT LIKELIHOODS FOR DIFFERENT MODES

The conditional probability distributions $p(y|x, m)$ are given in Table II. In terms of constraints, we assume that

there is a single resource pool of $R$ seconds to be used before all objects need to be classified. This number will also be varied across the experiments from 300 seconds to 700 seconds, to evaluate the bounds and algorithm performance for scenarios where the amount of sensor resources ranges from poor to rich.

In order to evaluate the utility of the lower bound, we compare the bound with the performance of two adaptive SM algorithms: a variation of Kastella's discrimination gain (DG) algorithm [5], which is a sequential algorithm for selecting the best sensor mode and target on the basis of maximizing the expected entropy reduction in the distribution of object type per unit sensor resource applied, and a dynamic scheduling algorithm (ADP) based on Lagrangian relaxation and POMDP approximations described in [7].

Each algorithm was simulated for 100 independent Monte Carlo runs using the same measurement outcomes to evaluate its average performance for three different levels of sensor resources: 300 seconds, 500 seconds and 700 seconds. The expected cost results were compared with the predictions of the lower bound. Table III includes the results for 300 seconds and 700 seconds of resources for six levels of missed detection (MD) costs. In this more complex case, the bound shows that there is room for improvement in both of the algorithms, although the performance of the algorithms is close to optimal for some of the conditions. For instance, when MD is close to 1, the costs of missed detections and false alarms is close, and policies such as maximizing information gain as measured by entropy are near-optimal. Similarly, the performance of the ADP is closer to the lower bound for limited sensor resources, as the limited look ahead approximation is closer to the actual optimal number of sensor actions per object.

| | 700 Seconds | | 300 Seconds | | | |
|---|---|---|---|---|---|---|
| MD | Bound | ADP | DG | Bound | ADP | Greedy |
| 1 | 1.6 | 1.58 | 1.91 | 4.61 | 9.7 | 9.17 |
| 5 | 4.5 | 4.46 | 6.75 | 15.66 | 17.03 | 18.62 |
| 10 | 6.5 | 6.49 | 9.87 | 19.56 | 21.18 | 20.71 |
| 20 | 8 | 8.25 | 14.87 | 21.67 | 22.38 | 22.11 |
| 40 | 10 | 10.01 | 23.05 | 24.18 | 24.53 | 24.91 |
| 80 | 11.25 | 14.6 | 29.85 | 26.38 | 26.38 | 30.5 |

TABLE III
PERFORMANCE OF SCHEDULING ALGORITHMS VS. BOUND

Figure 1 shows the results for the two algorithms and the lower bound for 500 seconds of sensing resource time. The results show that there is significant room for improvement in both policies: the discrimination gain (DG)algorithm fails to incorporate the relative values of different types of errors in its information seeking policy, and the ADP is conservative in that it does not use mixed policies and uses a limited lookahead, and thus can underutilize sensor resources.

## VII. DISCUSSION

In this paper, we have presented a lower bound for the achievable classification performance for a network of sensors with finite sensing resources. The approach is based on formulation of the adaptive sensing problem as a partially observed Markovian decision problem, which is then approximated by expanding the admissible decision space. This approximate formulation can be posed as an integer programming problem that has a separable dual formulation. A key result in establishing this separability is to show that the lower bound formulation can be solved in terms of a subset of sensing policies known as mixed local sensing policies, which are random mixtures of policies that select actions on each object based only on the past information collected on that object.

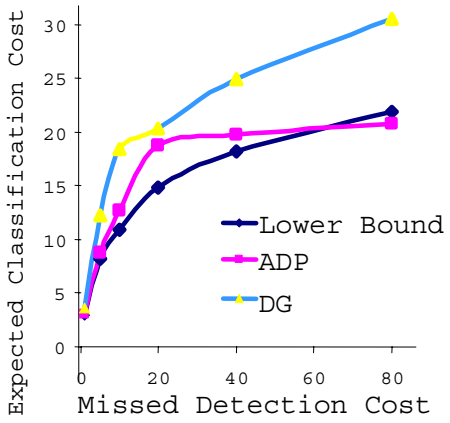


Fig. 1. Monte Carlo performance of algorithms and lower bound for 500 seconds of sensor resource.

We presented experimental results that compared the lower bound with the performance of two suboptimal adaptive sensing algorithms available in the literature. The experimental results established that the performance of both algorithms can be improved substantially in order to achieve the lower bound, and that the bound is tight in that the performance of the suboptimal algorithms is close to the predicted performance of the bound for several conditions.

In terms of sensor networks, the bound in this paper neglects the cost of communications as compared to the cost of active sensing. This is the case when sensors are in near vicinity of each other, and sensing requires active emissions by the sensors, so that the two-directional path loss is significant. In situations where communications also consume significant number of resources, the bound is optimistic, and would not be a good prediction for sensor network performance.

## REFERENCES


[1] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable Markov processes over a finite horizon" *Op. Res.*, V. 21, p 1071-1088, 1973.

[2] G. E. Monahan, "A survey of partially observable Markov decision processes: Theory, models and algorithms," *Mgmt. Sci.*, V. 28, p1-16, Jan. 1982.

[3] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Vols. I-II, Athena Scientific, Belmont, MA 1995.

[4] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA, 1999.

[5] K. Kastella, "Discrimination Gain to Optimize Detection and Classification,"*IEEE Trans. on Systems, Man and Cybernetics, Part A*, V. 27, No. 1, Jan. 1977.
[6] D. A. Castañón, "Optimal search strategies for dynamic hypothesis testing," *IEEE Trans. Sys., Man & Cybernetics*, v. 25, 1995.
[7] D. A. Castañón, "Approximate Dynamic Programming for Sensor Management," *Proc. 36th IEEE Conference on Decision and Control*, San Diego, CA, December 1997.
[8] D. A. Castañón and J. M Wohletz, "Model Predictive Control for Unreliable Dynamic Task Assignment," *Proc. 2002 Conf. Decision and Control*, Las Vegas, NV, Dec. 2002.
[9] G. Cohen, "Auxiliary problem principle and decomposition of optimization problems," *J. Opt. Theory and Appl.*, V. 32, 1980.
[10] W. S. Lovejoy, "A survey of algorithmic methods for partially observable Markov decision processes," *Annals of Operations Research*, v. 28, 1991.
[11] C. C. White, "Partially observed Markov decision processes: A Survey," *Ann. of Op. Res.*, V. 32, 1991
[12] M. L. Littman, A. R. Cassandra and L. Pack-Kaelbling, "Efficient dynamic programming updates in partially observable Markov decision processes," working paper, Brown University, Dec. 1995.
[13] A. R. Cassandra, *Exact and Approximate Algorithms for Markov Decision Processes*, Ph. D. Dissertation, Brown University, Providence, RI 1998.
[14] A. R. Cassandra, M. L. Littman and N. L. Zhang, "Incremental Pruning: A Simple, Fast Exact Method for Partially Observed Markov Decision Processes," *Proc. 13th Conf. Uncertainty in Artificial Intelligence*, Providence, RI 1997.
[15] M. R. Garey and D.S. Johnson,*Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, 1979.
[16] A. M. Geoffrion, "Lagrangian relaxation for integer programming," *Math. Prog. Studies*, v. 2, 1974.
[17] V. Krishnamurthy and R. J. Evans, "Hidden Markov Model Multiarm Bandits: A Methodology for Beam Scheduling in Multitarget Tracking," *IEEE Trans. Signal Processing*, V. 49, N. 12, Dec. 2001.
[18] P. C. Gilmore and R. E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem" *Operations Research*, V. 9, 1961.
[19] V. M. Demyanov and L. V. Vasilev, *Nondifferentiable Optimization*, Optim. Software, New York 1985.
[20] S. J. Benkoski, M. G. Monticino, and J. R. Weisinger, "A Survey of the Search Theory Literature," *Naval Research Logistics*, Vol. 38, No. 4, 1991, pp. 469-494.
[21] K. A. Yost, *Solution of Large-Scale Allocation Problems with Partially Observed Outcomes*, Ph. D. Thesis, Naval Postgraduate School, Monterey, CA, Sept. 1998.
[22] K. A. Yost and A. R. Washburn, "The LP/POMDP Marriage: Optimization with Imperfect Information," *Naval Research Logistics*, Vol 47, No. 8, 607-619, 2000.
[23] K. A. Yost and A. R. Washburn, "Optimizing Assignments of Air-to-Ground Assets and BDA Sensors," *Military Operations Research*, Vol. 5, No. 2, 77-91, 2000.
[24] R. Chen and G. L. Blankenship, "Dynamic Programming Equations for Discounted Constrained Stochastic Control, " *IEEE Trans. Automatic Control*," v.49, no. 5, May 2004.
[25] D. A. Castañón, "A Lower Bound on Adaptive Sensor Management Performance for Classification," CISE Report 2005-3, Boston University, January 2005.
[26] M. Schneider, "An Optimal Policy for Sensor Management with Symmetric Sensor Measurements," ALPHATECH working paper, August 2004.